Learning to de-anonymize social networks

Kumar Sharad

University of Cambridge
Computer Laboratory
Churchill College

November 2016

This dissertation is submitted for
the degree of Doctor of Philosophy

Learning to de-anonymize social networks

Kumar Sharad

Summary

Releasing anonymized social network data for analysis has been a popular idea among data providers. Despite evidence to the contrary the belief that anonymization will solve the privacy problem in practice refuses to die. This dissertation contributes to the field of social graph de-anonymization by demonstrating that even automated models can be quite successful in breaching the privacy of such datasets. We propose novel machine-learning based techniques to learn the identities of nodes in social graphs, thereby automating manual, heuristic-based attacks. Our work extends the vast literature of social graph de-anonymization attacks by systematizing them.

We present a random-forests based classifier which uses structural node features based on neighborhood degree distribution to predict their similarity. Using these simple and efficient features we design versatile and expressive learning models which can learn the de-anonymization task just from a few examples. Our evaluation establishes their efficacy in transforming de-anonymization to a learning problem. The learning is transferable in that the model can be trained to attack one graph when trained on another.

Moving on, we demonstrate the versatility and greater applicability of the proposed model by using it to solve the long-standing problem of benchmarking social graph anonymization schemes. Our framework bridges a fundamental research gap by making cheap, quick and automated analysis of anonymization schemes possible, without even requiring their full description. The benchmark is based on comparison of structural information leakage vs. utility preservation. We study the trade-off of anonymity vs. utility for six popular anonymization schemes including those promising k-anonymity. Our analysis shows that none of the schemes are fit for the purpose.

Finally, we present an end-to-end social graph de-anonymization attack which uses the proposed machine learning techniques to recover node mappings across intersecting graphs. Our attack enhances the state of art in graph de-anonymization by demonstrating better performance than all the other attacks including those that use seed knowledge. The attack is seedless and heuristic free, which demonstrates the superiority of machine learning techniques as compared to hand-selected parametric attacks.

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

It is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my dissertation has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University of similar institution except as declared in the Preface and specified in the text.

This dissertation does not exceed the regulation length of 60 000 words, including tables and footnotes.

Acknowledgments

First and foremost, I would like to thank my supervisor Ross Anderson without whom this dissertation would have not been possible. He helped me at critical junctures, provided encouragement and valuable feedback, I owe him much gratitude for whatever I managed to achieve at Cambridge.

I thank George Danezis for mentoring me during the initial stages of my PhD and teaching me how to do research. I thank Richard Clayton for helping me develop as a researcher; I can now fully appreciate the significance of sharing an office with him as a fresh PhD student and absorbing some of the wisdom on offer.

I thank Alastair Beresford for his guidance and help throughout my PhD; especially, as an examiner both for my first year report and final dissertation. I thank Emiliano De Cristofaro for generously agreeing to examine my dissertation and providing meticulous feedback.

I thank Lise Gough who went above and beyond in helping me come to Cambridge and then throughout my stay here. I could not have hoped for a more helpful and sincere person; it is due to her efforts that I was able to pursue a PhD at Cambridge. I thank Rebecca Sawalmeh for her ever helpful presence and making my stay at Churchill pleasant and comfortable.

I thank Juan Garay and Abhradeep Guha Thakurta for graciously hosting me at Yahoo Labs, Sunnyvale for a very productive internship. Those were by far the sunniest three months I have had since leaving India in 2009.

I thank the members of the Security Group and the Computer Laboratory, past and present, for their help and support including Ilias Marinos, Laurent Simon, Dimosthenis Pediaditakis, Marios Omar Choudary, Sheharbano Khattak, Rubin Xu, Dongting Yu, Bjoern Zeeb, Christian O'Connell, Alice Hutchings, Sophie Van Der Zee, Wei Ming Khoo, Timothy Goh, Jonathan Anderson, Markus Kuhn, Robert Watson and Frank Stajano.

I thank my sponsors Microsoft Research, EPSRC, Computer Laboratory, University of Cambridge and Churchill College for supporting my PhD studies at Cambridge.

Finally, I thank my family – *Ma*, *Papa* and *Didi* for their support throughout my education and their understanding even though I could not visit them as frequently as I would have liked to in the past few years. I dedicate this dissertation to them.

Contents

1 Introduction **13**

1.1 Chapter outline 13

1.2 Publications 15

1.3 Statement on research ethics 16

2 Background **17**

2.1 Privacy in high-dimensional datasets 17

2.1.1 Pitfalls of releasing private datasets 18

2.2 Privacy challenges in social networks 20

2.3 Anonymity loves company 21

2.4 Differential-privacy-based schemes 22

2.5 Clustering-based schemes 24

2.6 Perturbation-based schemes 25

2.6.1 k-Anonymity-based schemes 26

2.7 The adversarial model 29

2.8 De-anonymizing social networks 30

2.8.1 Seed-based attacks 30

2.8.2 Seedless attacks 33

2.9 Definitions 34

2.10 Summary 35

3 Automating social graph de-anonymization **37**

3.1 Introduction 37

3.2 Motivation: the D4D challenge 39

3.2.1 Data release and anonymization 40
3.2.2 Robustness of anonymization 41
3.2.3 Ad-hoc de-anonymization 42
3.2.4 Limitations of ad-hoc de-anonymization 43
3.3 Learning de-anonymization 44
3.3.1 De-anonymization: a learning problem 44
3.3.2 Decision trees and random forests 46
3.3.3 Specialized social graph features 48
3.3.4 Training and classification of node pairs 49
3.4 Evaluation 51
3.4.1 Experimental setup 52
3.4.2 Results: same training distribution 53
3.4.3 Results: different training distribution 55
3.4.4 Traditional de-anonymization task 57
3.4.5 Error analysis 59
3.4.6 Data sample sizes 61
3.4.7 Performance 61
3.5 Discussion 63
3.5.1 Is anonymization effective? 65
3.5.2 Improving de-anonymization 66
3.5.3 Choosing tree parameters 66
3.6 Summary 69

4 Benchmarking social graph anonymization schemes 71
4.1 Introduction 71
4.2 Quantifying anonymity in social graphs 73
4.2.1 The adversarial model 73
4.2.2 Graph generation 74
4.2.3 The classification framework 74
4.3 Evaluation and results: anonymity vs. utility 76
4.3.1 Random Sparsification (RSP) 79

4.3.2 Random Add/Delete (RAD) 80
4.3.3 Random Switch (RSW) 82
4.3.4 Random Edge Perturbation (REP) 85
4.3.5 k-Degree Anonymization (KDA) 87
4.3.6 1-hop k-Anonymization (1HKA) 90
4.4 Comparing social graph benchmarking schemes 94
4.5 Discussion 97
4.5.1 Risk of re-identification 98
4.5.2 Comparing anonymization schemes 99
4.6 Summary 103

5 The next generation of social graph de-anonymization attacks 105
5.1 Introduction 105
5.2 The de-anonymization landscape 106
5.2.1 Anonymizing social networks 106
5.2.2 Heuristics vs. machine learning 107
5.3 Evaluation 107
5.3.1 The adversarial model 108
5.3.2 Learning to de-anonymize 108
5.3.3 Unraveling anonymization 109
5.3.4 Datasets and implementation 111
5.4 Results 113
5.4.1 Mapping accuracy and coverage 113
5.4.2 Evolution of mappings 115
5.4.3 Error analysis 116
5.5 Comparison with the state of the art 121
5.6 Discussion 126
5.7 Summary 129

6 Conclusions 131

Bibliography 137

List of figures **155**

List of tables **159**

Chapter 1

Introduction

1.1 Chapter outline

The eminent Greek philosopher Aristotle noted that man by nature is a social animal, and this is evident from the huge popularity of social networks of all kinds. Being a part of such networks provides us a feeling of wellbeing and comfort as well as utility. Many people participate actively in social networks [1] and their experience gets richer the more information they share. The information contained in these networks ranges from public through private to downright secret; interconnectivity adds a novel dimension as it may reveal details unknown even to the individual themselves.

Social networks. We define social networks as networks where nodes represent people and edges represent relationships among them. These relationships are ties that are social in nature such as friendship, common interests, common workplace, professional relations etc. In general any social relationship which has some form of cost attached to it. In this dissertation we primarily focus on social networks which are formed as a result of first order interactions between people; some examples of such networks are Facebook, Flickr, Linkedin, Twitter etc. Such networks have similar properties even though they might be of very different nature; this allows us to group them together in a class for analysis. Social networks which depict the second-order interactions among people such as communication between data centers, road networks, telephone networks etc. also have some similarity to social networks involving people but these networks do not characterize human behavior as closely and definition of privacy is diluted in such a scenario, hence we concentrate on social networks involving people.

The anonymization of social graphs is becoming an important problem because of the growing secondary uses of social network data. It is imperative to understand the privacy guarantees we can expect from an anonymization scheme as it helps protect the data better. Data are often high-dimensional, content rich and very desirable for the research

community. Social network datasets have been a particular favorite of researchers and are shared by data holders fairly openly [2, 3]. Social network datasets are very helpful for studying a variety of human behavior such as mobility patterns, population growth, design of roads, crime hotspots, spread of epidemics, residential segregation, community structure etc. On the other hand such datasets can be very damaging if used unwisely. Releasing private datasets in countries struggling with political unrest could prove detrimental as it can be used for nefarious purposes. We discuss the case of Ivory Coast in detail in Chapter 3 where Orange decided to release call detail records for analysis. When data are collected from social networks that are inherently sensitive in nature such as political beliefs, LGBT groups, romantic interests etc. then the capacity for personal harm far exceeds the envisioned benefits. This also raises some ethical dilemmas – would it be fair to violate the privacy of a few individuals if there is social benefit in doing so? How should these people be chosen? Can societal benefits be achieved only at the cost of privacy? These are hard questions, solutions to which are essentially non-technical and require societal debate.

Data providers give an illusion of privacy by lightly anonymizing the data prior to release. Such measures are helpless against serious adversaries as they cannot hide second-order relations among data subjects. Yet so far, the research on social graph de-anonymization has been as ad-hoc as the anonymization techniques they attack. Hence, it is important to study whether such datasets can be anonymized before putting our faith in techniques promising anonymity. We study this via adversarial models which capture the capabilities of an attacker looking to defeat the anonymization of social graph datasets. Privacy of the individuals in a dataset can be breached in numerous ways, such as – discovering participation in a social network, exposure of social ties, learning political leanings, finding movie preferences etc. In isolation released datasets seldom pose a privacy risk however, the adversary often combines the released datasets to discover private information. Most of these attacks involve splicing datasets together; we define an adversarial model to capture these threats (see, § 2.7) and evaluate the likelihood of a privacy breach by designing efficient ways to do so. In this dissertation we systematize social graph de-anonymization attacks by taking a modular and principled approach to build a unified framework for evaluating and attacking graph anonymization schemes.

We make three primary contributions in this work. Staring from the design of a machine learning model to de-anonymize social graphs, we show how to use machine learning to benchmark anonymity schemes and finally mount an attack which surpasses the previous state of the art.

Automating social graph de-anonymization. Social graphs carry a lot of information some of which must be retained to preserve utility. In Chapter 3, we start with using the utility constraint to cast the problem of social graph de-anonymization as a learning task and build a model using machine learning techniques to accomplish it. We define

simple features based on neighborhood degree distribution to represent the nodes. The algorithm presented is flexible and amortizes the cost of de-anonymization in the face of changing anonymization schemes.

Benchmarking social graph anonymization schemes. Despite the many failed schemes to date, social graph anonymization schemes are widely proposed. Due to the diversity of adversarial models used and the complexities of capturing privacy leaks, there is no standard way to compare these schemes. Chapter 4 defines the learning task to create a framework which automates benchmarking of graph anonymization schemes under a common adversarial model. The framework does not require the description of the anonymization scheme but learns from examples provided. We also conduct a thorough analysis of six perturbation-based social graph anonymization schemes with varying levels of perturbation to assess its effect on graph utility. Our findings show that none of these schemes is fit for purpose if the perturbation is so light that the dataset's utility is preserved.

Mappings nodes across graphs. Finally, Chapter 5 proposes improvements to a number of heuristic-based attacks that map node pairs across intersecting social networks. Some attacks use known seed mappings for priming themselves while others are purely structure-based. We take a different approach by proposing a machine-learning based attack. Automating the selection of attack parameters using model training is shown to be far superior to handcrafted heuristics. Our attack is more successful than all other attacks and does not use any seeds or side information. Learning from pertinent examples not only automates the attacks but also radically improves them.

1.2 Publications

As a part of this dissertation I have published the following papers and posters, some in collaboration with other researchers, in peer reviewed academic conferences and workshops.

- Kumar Sharad. Change of guard: The next generation of social graph de-anonymization attacks. In *Proceedings of the 9th Workshop on Artificial Intelligence and Security*, AISec '16, 2016.
- Kumar Sharad. True friends let you down: Benchmarking social graph anonymization schemes. In *Proceedings of the 9th Workshop on Artificial Intelligence and Security*, AISec '16, 2016.
- Kumar Sharad and George Danezis. An automated social graph de-anonymization technique. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, pages 47–58, New York, NY, USA, 2014.

- Kumar Sharad and George Danezis. De-anonymizing D4D datasets. In *6th Workshop on Hot Topics in Privacy Enhancing Technologies*, HotPETs '13, 2013.
- Kumar Sharad and George Danezis. Privacy leak in egonets (poster). In *University of Cambridge Computer Laboratory 75th Anniversary Poster Competition*. Cambridge, UK, 24th April, 2013.

1.3 Statement on research ethics

All the experiments reported in this dissertation are performed using freely available and open social network datasets specifically published for research by the relevant organizations. We do not perform de-anonymization attacks on live social networks, scraped data, or previously unlinkable datasets.

Chapter 2

Background

In this chapter we look at the privacy landscape and issues concerning the release of high-dimensional datasets. We closely examine the externalities of anonymizing and de-anonymizing such datasets with a focus on online social networks. We take a look at how privacy attacks and defenses have co-evolved in social graphs. This chapter forms the backdrop for the work presented in subsequent chapters.

2.1 Privacy in high-dimensional datasets

As much of our lives have become digitized, gathering private and personal data has never been easier. The collection and documentation of our digital lives has manifested itself in various forms ranging from product ratings, movie preferences, click patterns, search history, geo-location data, medical records, browsing profiles, shopping preferences, communication patterns, travel histories, media preferences and, especially, social networks. Much of this data is high-dimensional and feature-rich thus providing a valuable opportunity to mine human behavior. The demand for such data among analysts and researchers is very high, and to fill this gap data providers often release data for the public good. However, releasing such data poses a serious threat to the privacy of the data subjects [3–5]. The literature suggests that it may not be possible to effectively anonymize high-dimensional data without destroying its utility [6]; this is particularly true for social graphs due to entities being interlinked. Privacy of individuals in the dataset is seldom a priority in such releases and is viewed as an obstruction to analysis. Until recently very little was done to protect the privacy of individuals whose data were published, however over the past decade data providers have started taking privacy a bit more seriously and have resorted to anonymizing data before publication. Research shows that effectively anonymizing high-dimensional data is a very hard problem [7–22].

Data providers primarily employ data anonymization strategies to protect themselves from being held accountable for privacy invasion of the individuals. In the USA, providers use

the basic legal definition of *Personally Identifiable Information* (PII) to frame privacy rules about PII such as those in the *Health Insurance Portability and Accountability Act*[1] (HIPAA) try to provide some guidance about data anonymization, but the concept is flawed, especially for high-dimensional datasets where it completely fails to provide any acceptable level of privacy [23–25]. In the UK, providers rely on the fact that the transposition of the Data Protection Directive is flawed, leading to a defective definition of personal data that lets them claim data to be anonymous even when recipients can easily re-identify it[2,3]. Law is yet to catch up with the privacy research in this area and the gap leaves much scope for manipulation and interpretation of results which is exploited by data providers. Catastrophic privacy breaches have been caused by data releases where the data providers have kept repeating the same mistake of assuming that data which appear to be anonymous are actually so (as we shall discuss in the next section). The lessons are being learned only slowly due to the lack of legal accountability.

2.1.1 Pitfalls of releasing private datasets

Traditional approaches to protecting privacy in databases have long become obsolete. This shift has happened primarily due to modern databases having a large number of columns or attributes, popularly described as high-dimensional. Due to the high number of attributes each data point (often representing an individual) is unique or almost so. Beyer *et al.* [26] show that under a broad set of conditions, the distance to the nearest data point approaches the distance to the farthest data point with the increase in dimensionality. Hence it is very challenging to group data points together in the hopes of becoming inconspicuous in the crowd. Such attempts come at a huge cost to the utility that was the primary goal of publishing the data.

Despite this, data providers continue to test the boundaries with adventurous ways of releasing personal data. One of most famous privacy fiascoes is attributed to AOL[4]. In August 2006 AOL published anonymized search logs containing 20 Million queries collected from over 657 000 of its users over a three-month period. The data were released to the public and was meant to be used for research purposes. Two New York Times journalists were able to identify *user no. 4417749* just by looking at the content of the queries which contained a lot of unsanitized personal information. On top of violating the privacy of a great number of users the incident caused AOL a great deal of embarrassment and also led to a class action lawsuit[5]. CNN recounts this episode as number 57 in their list of *101*

[1] https://aspe.hhs.gov/report/health-insurance-portability-and-accountability-act-1996

[2] https://www.gov.uk/government/publications/iigop-report-on-caredata

[3] https://www.dataprotection.ie/docs/EU-Directive-95-46-EC-The-Recitals-Page-1/90.htm

[4] http://query.nytimes.com/gst/abstract.html?res=9E0CE3DD1F3FF93AA3575BC0A9609C8B63

[5] http://www.cnet.com/news/aol-sued-over-web-search-data-release/

Dumbest Moments in Business[6].

Narayanan and Shmatikov [3] demonstrated the challenge that availability of side information poses to the privacy of a published dataset. They presented statistical de-anonymization attacks against high-dimensional micro-data. The attacks can successfully de-anonymize individuals even with imperfect knowledge and noise in the data. Their paper demonstrated the efficacy of the proposed attacks by identifying subscribers in anonymized Netflix[7] movie ratings, using Internet Movie Database[8] (IMDb) ratings as background knowledge.

We discovered similar privacy issues with the de-anonymization of social networks. Orange introduced the *Data for Development* (D4D) challenge[9] in the Ivory Coast to support research on socio-economic development. They shared an anonymized version of call detail records with researchers to facilitate the study. We found a number of issues with their anonymization procedure [27], including serious privacy leaks due to structural attacks; Chapter 3 describes these issues in further detail. The possible damage was restricted because the data were not publicly released, but carefully shared with hand-picked research teams only after signing a non-disclosure agreement.

In March 2014, following a freedom of information request, a complete dump of historical trip and fare logs from the New York City taxis was published[10]. The data contained more than 173 Million individual trip records, each containing detailed timestamped movement history and taxi charges. Each trip also included the hack license number and medallion number which were anonymized by replacing them by their MD5 hashes[11]. Such hash functions are only useful in anonymization if there are a very large number of possibilities for the hashed inputs. However, there are only about 22 Million possibilities for both the hack license and medallion numbers, which can be hashed in minutes. Once the hashes were known it was trivial to look up the correct number, which could then be correlated with other datasets to reveal the drivers' identity, their complete movement pattern and the wages they earned. The anonymization attempt collapsed due to poor use of cryptography.

Such instances point to the serious issues with claims of social benefit from publishing private datasets. Once the data are released they live on and every additional release shrinks the privacy horizon further. The datasets seldom exist in isolation and any interaction makes the adversary stronger. However, parties involved in data release live in a bubble and almost never consider the effect of each others' actions. Attacks only ever get better thus making privacy breaches ever more likely with every new data release.

[6] `http://money.cnn.com/galleries/2007/biz2/0701/gallery.101dumbest_2007/57.html`
[7] `https://www.netflix.com`
[8] `http://www.imdb.com`
[9] `http://www.d4d.orange.com/`
[10] `https://medium.com/@vijayp/of-taxis-and-rainbows-f6bc289679a1`
[11] `https://tools.ietf.org/html/rfc1321`

Social networks being high-dimensional are even more challenging to anonymize. They present a variety of privacy issues which are unique; we take a look at some of them in the next section.

2.2 Privacy challenges in social networks

Preserving privacy has been an ongoing tussle since the advent of online social networks. Effectively anonymizing network data is challenging, and balancing privacy and utility is even harder [28]. In addition, social networks can be the target of a wide variety of attacks. Often private attributes of an user can be leaked just by observing public attributes such as friendship and group membership [29].

Chew *et al.* [30] highlight the problem of inference control in social graphs. The problem manifests itself via social graphs in several forms such as publication of activities, unwelcome linkage of personae from various sources and merging of social graphs – all of which erode privacy. The authors study the social network landscape and evaluate how privacy breaches are enabled by popular social networks; ways to mitigate these risks are also proposed.

Felt and Evans [31] propose mitigating the risk of malicious data harvesting in social networks by third-party APIs through a privacy-preserving design. Third parties are restricted to only an anonymized social graph and an abstracted version of the user data. The suggested privacy policy can be implemented with the help of the platform owner and the study argues that the recommended level of access is sufficient for almost all applications.

Lucas and Borisov [32] aim to protect the information published by the user of a social network from the platform owner itself via cryptographic techniques. The architecture presented makes active attacks, even though possible, expensive for the platform owner. Encryption ensures that the servers never store information in cleartext while supporting many-to-many communication and ease of accessibility. The platform is used to manage encryption keys and social relationships. Analysis of an implemented Facebook[12] prototype shows reasonable balance between privacy and usability.

Singh *et al.* [33] present a framework for building privacy-preserving social networking applications without sacrificing the functionality of the platform. The framework uses information flow models to restrict harvesting of user data by third parties.

Kerschbaum and Schaad [34] present a mechanism for privacy-preserving social network analysis to facilitate collaboration of criminal investigators without plaintext data exchange. The authors propose protocols using encryption to compute graph metrics without releasing

[12] `https://www.facebook.com`

the original social graph. The technique allows selective disclosure of social network information, thus augmenting the social graph's view of the investigator.

Frikken and Golle [35] propose cryptographic techniques to assemble pieces of a distributed social graph without any of the data holders revealing the identity of their users. The assembled graph is isomorphic to a perturbed version of the underlying graph and restricts the adversary from learning a true isomorphism between the published and the underlying graph.

Korolova *et al.* [36] present an attack where an adversary subverts user accounts to gain information about their neighborhood and collates it to obtain a global view of the network. They present several strategies and an analysis of the fraction of users to subvert depending upon the choice of users and the user neighborhood lookahead visible for the attack to be successful. The results show that a large lookahead has a huge potential for privacy breach and should be controlled to provide a balance between utility and privacy.

These issues broadly highlight the privacy challenges related to social networks. We now focus our attention to the challenges of preserving privacy in the face of social network data release.

2.3 Anonymity loves company

Latanya Sweeney [37] proposed the notion of k-anonymity to anonymize relational databases. It provides a formal protection model to safeguard the privacy of individuals whose data is released. A released dataset provides k-anonymity protection if the information for each individual contained in the data release cannot be distinguished from at least $k-1$ other individuals whose information has also been released. Ensuring k-anonymity is expensive in general [38] and suffers from a homogeneity attack: a set of individuals might be k-anonymous yet still share the same value of a sensitive attribute, thus causing a privacy breach. Background-knowledge attacks could also be mounted where the adversary can leverage the relation between quasi-identifier attributes and sensitive attributes to narrow the range of possible sensitive attribute values.

Machanavajjhala *et al.* [39] propose l-diversity to overcome the shortcomings of k-anonymity. An equivalence class is said to be l-diverse if there are at least l *well-represented* values for the sensitive attribute. A dataset is considered l-diverse if all the equivalence classes of the dataset are l-diverse. This is similar to the approach taken by p-sensitive k-anonymity proposed by Truta and Vinay [40]. A dataset satisfies p-sensitive k-anonymity if it satisfies k-anonymity and for each group of tuples with the identical combination of key attribute values that exists in dataset, the number of distinct values for each sensitive attribute occurs at least p times within the same group.

Li *et al.* [41] further refine l-diversity to propose t-closeness where the distance between

the distribution of sensitive attribute in the equivalence class and the distribution of the attribute in the whole dataset is bounded by the threshold t. Domingo-Ferrer and Torra [42] show that neither k-anonymity nor its enhancements are completely successful in protecting privacy while preserving data utility.

The proposed k-anonymity and related techniques have been extended beyond relational databases and applied to social graphs as well, though they are even less potent due the high number of dimensions. Privacy in social graphs can be sought in an interactive or non-interactive setting. The interactive approach of differential privacy provides rigorous privacy guarantees in certain cases but is challenging to set up and requires constant monitoring by the data holder. While non-interactive social graph anonymization schemes are easy to set up and require no monitoring, they come without any provable privacy guarantees; such schemes mainly fall under two categories [43–45] – clustering-based schemes and perturbation-based schemes. We discuss these approaches in the next sections.

2.4 Differential-privacy-based schemes

Dwork *et al.* [46, 47] proposed differential privacy as a mechanism to provide provable privacy guarantees while allowing access via queries to a statistical database. Such mechanisms evolved in response to catastrophic privacy breaches caused by simplistic anonymization of published data [48]. Differential privacy has gradually gained in popularity with a sizable body of research exploring its applications [49–60]. However, such systems are non-trivial to set up and the privacy guarantees come at the cost of maintaining an interactive system to answer queries which are tightly controlled and include added noise to mask the true response. Differential privacy guarantees that an adversary querying a statistical database learns the same information about an individual irrespective of its presence in the database. Protecting privacy even with strong guarantees is challenging as definitions can break down due to improper assumptions [61].

Frameworks have been proposed to mold differential privacy to develop rigorous privacy definitions for specific data sharing needs [62]. Attempts have also been made to extend the notion of differential privacy to social networks [63–66]; the task here is more challenging due to the records being related to each other in contrast to a traditional statistical database which assumes independence of records. Most approaches designed to protect privacy in social graphs aim to provide either *edge differential privacy* [67–70] – protecting the presence of a relationship in a graph or the much stronger *node differential privacy* [71, 72] – protecting the presence of an individual along with all their relationships. Differential privacy is essentially an interactive technique but it has been used to suggest non-interactive approaches. Sala *et al.* [68] propose *Pygmalion*, a differentially private graph model to protect edge privacy. The model takes a graph and a pre-defined differential privacy guarantee and generates a structurally similar synthetic graph using

degree correlations. The approach provides a privacy guarantee comparable to the purely differential privacy case with much less noise. The authors aim to achieve a middle ground between the anonymized release of data and a tightly-controlled query-based system. The technique impacts the utility of the graphs and has limited applicability without advance knowledge of data processing needs. Additionally, the incorrect estimation of the privacy parameter exposes the data to structural graph de-anonymization attacks, while choosing the parameter too aggressively destroys all utility.

Wang and Wu [69] also employ degree correlation to provide edge differential privacy in graph generation. The technique enforces differential privacy on graph model parameters learned from the original graph; synthetic graphs are generated subsequently using the graph model with private parameters. Xiao *et al.* [70] use an estimate of the connection probabilities between the graph vertices to infer the structure of the graph in a differentially private manner with reduced noise. Proserpio *et al.* [73, 74] present a data analysis platform *wPINQ*, which non-uniformly scales down the contribution of challenging records instead of scaling up the magnitude of noise in differentially private computations. The authors show that their technique produces higher accuracy by circumventing the worst-case requirements of differential privacy and can improve results of graph analysis.

Privacy definitions that are more suited to social graphs have also been suggested, Gehrke *et al.* [75] propose a zero-knowledge based definition strictly stronger than differential privacy and particularly suited to modeling privacy in social networks, the authors demonstrate the computation of a variety of statistics under the proposed notion.

Approaches have also been proposed that use the interactive query based mechanism inspired by differential privacy but without any provable privacy guarantees. Such approaches are generally ad-hoc and typically provide an interface for data analysts to query. The queries aim to provide *safe* answers but do not tightly control the information revealed; this reduces the granularity and dimensionality of the information and limits the scope of attack. de Montjoye *et al.* [76] propose openPDS – a personal metadata management framework that allows individuals to collect, store, and give fine-grained access to their metadata to third parties. The platform allows services to get answers to queries which are computed on an individual's metadata instead of trying to anonymize it. Revealing pre-computed results reduces the scope of privacy breach without publishing the dataset. The methodology does provide a compromise between pure differential privacy mechanism and publishing the entire database and might be worth considering depending upon the sensitivity of the dataset.

In summary, differential-privacy approaches are constrained due their complexity, their limited ability to handle general computation scenarios, and the reduced accuracy due to lack of data application knowledge which is seldom known beforehand; but they do open up an interesting new avenue for privacy protection research. Interactive mechanisms which are easier to implement also exist but come without any provable privacy guarantees

and should be used prudently.

2.5 Clustering-based schemes

Clustering-based graph anonymization schemes release aggregate graph information instead of the raw graph. We take a brief look at some of them in this section.

Zheleva and Getoor [77] consider the case of social graphs where nodes representing individuals have multiple types of relationships among them, some of which are sensitive. The relationships are represented by multiple edges between a pair of nodes; thus the social network is a multigraph. The authors assume that the nodes are clustered into equivalence classes based on their attributes. A number of measures are suggested to hide sensitive relationships, all of which include removal of sensitive edges. Additional privacy is provided by either deleting a fraction of non-sensitive edges, releasing non-sensitive edges among equivalence classes, deleting a fraction of non-sensitive edges among equivalence classes and deleting all edges. Multigraphs are not a very common way to represent social networks and the suggested measures have limited scope.

Cormode *et al.* [78] present a technique that perturbs the mapping from entity to nodes using safe groupings while keeping the structure of the graph intact.

Hay *et al.* [79] propose preserving privacy by grouping nodes into partitions. The published data include the number of nodes in each partition and density of edges that exist within and across partitions. This technique reduces the granularity of data by clustering nodes into supernodes and edges into superedges. The authors suggest sampling a random graph from the published data for graph analysis.

Campan and Truta [80] present a clustering mechanism similar to Hay *et al.* [79], but propose preserving privacy under one extra complication – when the nodes have attributes. They present ways to generalize categorical attributes based on predefined hierarchies and numerical attributes using intervals. The generalizations are carried out at cluster level after splitting the nodes into clusters.

Bhagat *et al.* [81] propose using label lists to protect the privacy of node attributes and partitioning nodes into classes to protect against structural attacks. Instead of completely removing node identifiers they suggest replacing them with a list of identifiers among which one is its true identifier, the graph structure is left intact. The paper claims that choosing the list carefully prevents the attacker from guessing the true identifier with high probability. Such anonymization does not protect against structural attacks and the scenario seems limited in scope. The authors follow the approach of Hay *et al.* [79] and Campan *et al.* [80] to guard against structural attacks by partitioning the nodes into classes and only releasing the interaction between classes.

Clustering nodes and edges of a social graph is a useful way of publishing summarized results. It provides a certain amount of privacy with limited data utility as the published data is preprocessed. Summarizing information does away with the possibility of granular analysis and only provides an aggregate view. Perturbation-based schemes fare better in terms of data utility, and we discuss them next.

2.6 Perturbation-based schemes

Instead of releasing aggregate information, perturbation-based schemes introduce imperfections in the social graph before publishing it. The primary goal is to deter graph-structure-based de-anonymization attacks. Certain schemes appear frequently in the anonymization literature; they all involve deleting/adding edges in various ways. Some of them are:

- Random Sparsification (`RSP`) [5, 82] – This scheme deletes a fraction of graph edges at random.
- Random Add/Delete (`RAD`) [82–84] – The graph is perturbed by deleting a number of edges followed by introducing the same number of non-edges at random, this preserves the total number of edges in the graph.
- Random Switch (`RSW`) [82, 84–86] – This scheme involves randomly selecting two edges and switching them across node pairs which are not already connected. As seen in Figure 2.1 the edges (a, b) and (c, d) are transformed to either (a, c) and (b, d) or (a, d) and (b, c) given that those edges do not already exist. The number of edges and the individual node degrees are preserved by this scheme.

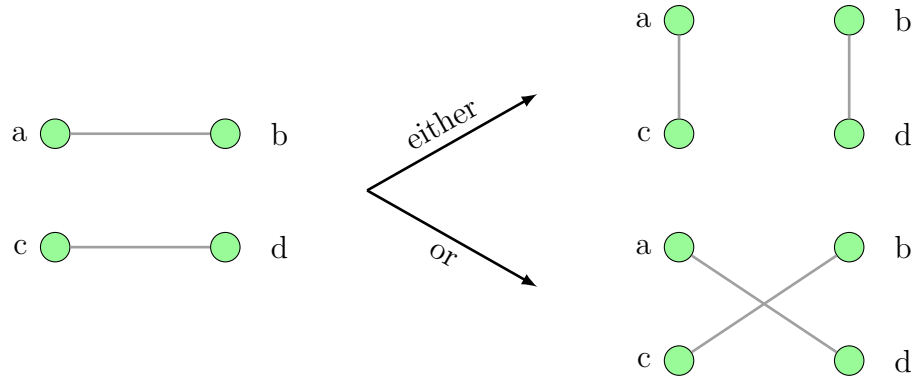


Figure 2.1: Random Switch

Ying and Wu [84] present a scheme to preserve the spectral properties of the graph which are often damaged by random perturbation. To this end, spectrum-preserving versions of `RAD` and `RSW` are proposed. The authors consider an adversary with only the knowledge of target node degree, such an adversary is weak and limited. The anonymity provided

is measured using a posteriori analysis of a link being discovered in the original graph after observing the perturbed graph. The anonymity provided by the spectrum-preserving approach is shown to be at most as good as `RAD` and `RSW`, which (as demonstrated in Chapter 4) are pretty weak to begin with. Sacrificing more privacy to enhance utility might make the scheme completely unusable for privacy protection. The paper only tries to mitigate a particular type of subgraph attack [87] and is inefficient for large graphs.

Liu *et al.* [88] propose ways to preserve edge-weight privacy of a released graph. The paper presents two algorithms. The first algorithm perturbs the edge weights by introducing Gaussian noise to preserve the length of the perturbed shortest paths. The second algorithm follows a greedy approach to preserve some of the shortest paths after perturbation as well as keep the perturbation in length to a minimum. The scheme covers a very narrow set of problems and is limited in scope.

Xue *et al.* [89] (Random Edge Perturbation – `REP`) suggest a scheme where the graph is perturbed by removing a fraction of edges and adding the same fraction of non-edges. Their aim is to defeat the narrow set of structural attacks presented by Backstrom *et al.* [87] where the attacker launches active (sybil) and passive attacks by searching patterns in sub-graphs to learn relationships between pre-selected target individuals. The authors claim that knowing the perturbation factor allows the estimation of important graph metrics. However, adding huge numbers of non-edges greatly warps the graph, which is then of little use (even with the knowledge of the perturbation factor which authors claim to be only known to the legitimate data recipient).

Mittal *et al.* [90] propose a random-walk approach to rewire edges in the graph to protect link privacy. The scheme provides a good balance between utility preservation and privacy achieved. The authors analyze link privacy based on Bayesian formulation of the adversary's priors as well as a risk-based measure which does not assume any prior knowledge by the adversary. A formal treatment of perturbed graph utility and its relation to privacy is also provided.

Liu and Mittal [91] propose LinkMirage to protect link privacy between labeled vertices. The authors propose clustering static and dynamic graphs into communities and then perturbing the inter-cluster and intra-cluster links to protect privacy. Clustering in dynamic graphs finds community structures in evolving graphs by simultaneously considering consecutive graphs; subsequently changed communities are perturbed to keep the perturbation minimal. Their approach allows high level of privacy in the perturbation process while preserving graph utility.

2.6.1 k-Anonymity-based schemes

Li *et al.* [92] show that k-anonymity based schemes fail to provide sufficient protection against re-identification. However, they can be preceded with a random sampling step to

provide levels of privacy guaranteed by differential privacy with reasonable parameters. k-anonymity provides some desirable properties; however, in its pure form it is known to be of little use in protecting high-dimensional datasets [6]. This has not prevented people from proposing a multitude of k-anonymity based graph anonymization schemes. Such schemes show a gradual progression – starting with making the degrees of nodes k-anonymous followed by 1-hop neighborhood to 2-hop neighborhood. The schemes that target neighborhoods aim to modify the sub-graph around each node in such a way that it becomes indistinguishable from $k-1$ other nodes. Such schemes are computationally very expensive and some are known to be NP-hard [93, 94]. Also, they are particularly damaging to the graph's overall utility; the anonymization strategy makes it hard to analyze community structure and related properties. We describe some of the schemes which fall under this sub-category in the next paragraphs.

Liu and Terzi [95] (k-Degree Anonymous – `KDA`) propose to make the graph k-degree anonymous. They model the attacker's prior knowledge as the degree of a target node and present a perturbation scheme that ensures that there are at least k nodes of any given degree. The authors present various algorithms to achieve k-degree anonymization. The proposed adversarial model is weak and has limited scope. Knowledge of the node degrees also allows the attacker to sketch attacks based on their distribution, but the authors do not discuss this and constrain the attacker to only use exact degrees.

Zhou and Pei [96] (1-hop k-Anonymous – `1HKA`) present a scheme to make the nodes in a social network k-anonymous with respect to their 1-hop neighborhood. The authors assume that the adversary only has knowledge of 1-hop neighborhood of the target node. The adversarial model is weak as attacks by much stronger adversaries already exist [5]. The problem is shown to be NP-hard and a greedy approach is proposed instead of finding the optimal solution. The utility of the anonymized graphs is not very encouraging and is limited in scope. The experiments are conducted on sparse graphs with low average degree which already require noticeable increase in the number of edges, the number of edges inserted and computational complexity are likely to rise significantly with increase in average node degree.

Thompson and Yao [97] propose schemes similar to `KDA` and `1HKA` to achieve k-degree anonymity and 1-hop k-anonymity. Their scheme relies on clustering nodes first and then anonymizing the node clusters by adding and deleting edges to conform them to the anonymity level sought. The scheme is not optimal and uses heuristics to cluster nodes.

Zou *et al.* [94] propose an anonymization scheme based on graph isomorphism. They modify the original graph by inserting vertices and edges such that each node has at least $k-1$ automorphisms. Such a strategy limits the probability of node re-identification to $\frac{1}{k}$. To achieve this, first, the graph is partitioned into blocks of sub-graphs, these blocks are then grouped into groups of size k. Each of the blocks in these groups are made isomorphic, the algorithm also considers the edges linking the blocks. The authors acknowledge that

finding optimal edge insertions to make blocks in a group isomorphic is NP-hard, and they propose heuristics to get around this problem. Finding the optimal graph partition is also shown to be NP-complete. The approach is marred by several critical problems which make the scheme unusable. The number of edges inserted is bounded by $(k-1) \cdot E$ (where E is the number of edges in the original graph), the paper does not provide a tighter bound and even for modestly high k the graph would be too noisy [82, 83]. Utility is analyzed using small graphs of low average degree at $k = 10$. Results on such graphs cannot be generalized and they deteriorate significantly at $k = 20$ which is fairly low. Additionally such techniques make it very hard to study community structure.

Cheng *et al.* [93] aim to protect against two kinds of attacks in anonymized social networks, *NodeInfo* – identifying information related to a node and *LinkInfo* – identifying relationship between a node pair. To this end they define the notion of k-security such that an adversary armed with any kind of structural knowledge cannot launch these attacks with a success probability of greater than $\frac{1}{k}$. The authors show that the problem is NP-hard. They subsequently provide a solution using k-isomorphism by splitting the original graph into k isomorphic disjoint sub-graphs such that they are pair-wise isomorphic, this is shown to be both sufficient and necessary for protection. The generated sub-graphs are meant to be a sample of the graph, but an analysis to confirm whether this sample is representative is missing. The *LinkInfo* attack is particularly notorious to protect against as it can be completely successful despite nodes being k-anonymous. This is identical to the problem that l-diversity [39] tries to solve. Anonymization using k-isomorphism is limited at providing privacy. The results presented are based on analysis of small graphs (largest graph has 20 000 nodes) with very low average degree (≈ 5) using small values of k. This does not at all represent real-world graphs and the results cannot be generalized. Even if we overlook this weakness, the scheme does not scale and cannot handle large graphs due to exponential runtime. Utility is measured by comparing path lengths and degree distribution of the original and perturbed graphs. Such analysis does not account for distortion of community structure, which is significant for the proposed scheme.

Wu *et al.* [98] propose k-symmetry to obscure node identities. The graphs are anonymized by forcing each node to have at least $k-1$ structurally equivalent counterparts to mask their identity irrespective of the structural knowledge that the adversary may possess. The authors achieve this goal by introducing new edges and vertices. A backbone sampling is also proposed to extract the core structure from the modified graph to compute global graph metrics. The suggested approach has several serious flaws. Firstly, the authors assume that the anonymization algorithm is provided with the list of vertices that are isomorphic to each other so that the algorithm can extend the size of the list to at least k. Computing this list is far from trivial and is known to be polynomially equivalent to the *Graph Isomorphism* problem [99]. The authors get around this hurdle by using `Nauty`[13]

[13] `http://cs.anu.edu.au/~bdm/nauty/`

to compute approximate lists. This does not scale well and has problems handling even graphs with as few as 20 000 nodes. The experiments conducted are in line with this and the largest graph used has merely 4 213 nodes, casting serious doubts on the applicability of the findings. Secondly, the anonymization algorithm has a time complexity of $\mathcal{O}(k^2 \cdot n^2)$ where n is the number of nodes in the graph, which is rather expensive. For real world graphs with tens of millions of nodes and moderately sized $k \approx 100$, the cost is very high. Lastly, the results presented claim to preserve global graph metrics to a large extent; however, the size of graph used for analysis is too small (just a few thousand nodes, as noted), with low average node degree, and values of k in the range of 5 to 10 which is far too small. Choosing higher k values would skew the graph by a huge factor; k-symmetry is a huge hindrance to allowing any local level analysis. Also, introducing vertices is specially damaging as such a graph no longer represents the true reality beyond preserving some global metrics which can be similar for graphs which are otherwise completely different.

As k-anonymization guarantees get stronger the complexity rises exponentially and we run into NP-hard problems. Even if efficiency is disregarded, the guarantees provided require suppressing huge amounts of information [6] thus rendering the data useless. In Chapter 4 we analyze a representative sample of six anonymization schemes – `RSP`, `RAD`, `RSW`, `REP`, `KDA` and `1HKA` in detail and compare them with the base case when graphs are not anonymized. The analysis studies the true marginal anonymity achieved purely due to the anonymization scheme employed and its effect on utility.

2.7 The adversarial model

Adversaries often combine datasets to launch attacks [3, 5, 100–104]. This dissertation uses the well-studied and widely used adversarial model originally proposed by Narayanan and Shmatikov [5, 102] to evaluate our attacks presented in Chapters 3 and 5. The model assumes that the adversary has access to an auxiliary graph which is used as side information to re-identify individuals in a sanitized graph [105]. Data holders often choose to sanitize the graphs prior to publishing them by removing the identifying information of individuals; usually noise is also introduced by manipulating edges.

We simulate this by sampling overlapping graphs from a large social graph. We take a real world social network $G = (V, E)$ and randomly partition the set of nodes V into two subsets V_1 and V_2 with an overlap α_V. The overlap between the node sets is measured by the Jaccard Coefficient, defined as:

$$JC(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{2.1}$$

where X and Y are sets at least one of which is non-empty. An overlap of α_V is obtained by randomly partitioning V into three subsets V_A, V_B and V_C with sizes $\frac{1-\alpha_V}{2} \cdot |V|$, $\alpha_V \cdot |V|$

and $\frac{1-\alpha_V}{2} \cdot |V|$ respectively and setting $V_1 = V_A \cup V_B$ and $V_2 = V_B \cup V_C$. Finally, we create two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ as node induced sub-graphs of G using vertex sets V_1 and V_2.

Further noise is injected through random sparsification (RSP, see § 2.6) – edges between nodes in G_1 and G_2 are deleted at random to produce the anonymized auxiliary and sanitized graphs $G_{aux} = (V_{aux}, E_{aux})$ and $G_{san} = (V_{san}, E_{san})$ respectively. This is done by making two copies of the edge set E and independently deleting edges at random from each copy. The two copies are then projected on to V_1 and V_2 to obtain E_{aux} and E_{san}. Deleting a fraction β of edges from each copy produces a fraction of $(1-\beta)^2$ common edges and a fraction of $1-\beta^2$ present in at least one copy. Thus the edge overlap measured as Jaccard Coefficient is, $\alpha_E = \frac{(1-\beta)^2}{1-\beta^2}$, to obtain this edge overlap a fraction $\beta = \frac{1-\alpha_E}{1+\alpha_E}$ of edges must be deleted from each copy of E. This overlap is only among the edges of the subgraph common to G_{aux} and G_{san}; the edge overlap for the entire graph is much lower. We refer to the vertex sets of G_{aux} and G_{san} as V_{aux} and V_{san} respectively, note that $V_{aux} = V_1$ and $V_{san} = V_2$. The nodes and edges are stripped of all identifiers, we only consider the structure of the graphs in further discussions. Deleting edges is popular and is in fact one of the best ways to introduce noise, as it produces an even degradation of graph features while providing anonymity superior to many other schemes, as will be shown in Chapter 4.

2.8 De-anonymizing social networks

Graph de-anonymization attacks broadly fall into two categories – seed-based and seedless. Seed-based attacks use known mappings across auxiliary and sanitized graphs referred as seeds to prime the de-anonymization process while seedless attacks proceed without the prior knowledge of any mappings. Some attacks augment other attacks to enhance their performance; Nilizadeh *et al.* [106] (NKA) exploit the community structure of social networks to augment de-anonymization. The authors propose a divide-and-conquer approach to de-anonymize nodes is two stages. The first stage maps the communities across graphs, followed by mapping users within the communities. Experiments on large real world social networks demonstrates that community-aware network alignment improves performance in the presence of noise and a low number of seeds.

2.8.1 Seed-based attacks

Srivatsa and Hicks [107] (SH) splice mobility traces with social network data to de-anonymize mobile users using common friends. The mobility traces are cast as social graphs to facilitate mapping with a related graph. The authors present a two step process; the first step identifies landmark nodes which are used to extend the mapping in the second

step using node similarity heuristics. Node centrality is used to identify landmark nodes in both graphs; these nodes are mapped by trying all possible combinations and selecting the most likely mappings using a goodness-of-fit measure. The mapping process is highly inefficient and can only handle toy problems taking about seven hours for a 125 node graph. The biggest graph used for evaluation only has 125 nodes and datasets have been hand curated for the study which raises concerns about the results' general applicability.

Narayanan and Shmatikov [5] (`NS`) present a two-phased attack to map nodes across overlapping directed graphs using graph topology. The attack presented is self-reinforcing and feedback-based which provides the adversary more auxiliary information as nodes are re-identified. The algorithm begins with re-identification of seed mappings in the first phase which are then propagated to expand the mappings in the second phase. The attack requires enough seeds of high degree and with specific structural properties. The propagation step utilizes the seeds and topological information to discover new mappings, the process is iterative and large scale re-identification is possible under the right circumstances. Running the two-phase attack is expensive even with the assumption of error free seed mappings, the time complexity of the proposed algorithm is $\mathcal{O}((e_1+e_2)\cdot d_1 \cdot d_2)$ where e_1 and e_2 are the number of edges and d_1 and d_2 are the maximum degrees in the two graphs. The experiments conducted on real world social graphs establishes the feasibility of the proposed attack.

Narayanan *et al.* [4] de-anonymize the Kaggle dataset released for link prediction using a pre-crawled version of the same dataset. Their work combines de-anonymization [5] and link prediction using random forests, however, the de-anonymization phase does not use any machine learning. Random forests are used to predict those links which pure de-anonymization could not uncover. Additionally, their work is based on availability of ground truth to mount de-anonymization attack on a dataset which is not adversarial. Cukierski *et al.*[14] got good results just using pure random forests for link prediction, rather than de-anonymization, for the same dataset.

Backstrom *et al.* [87] (`BDK`) were the first to present structural attacks on anonymized social networks. Both active and passive attacks were considered to de-anonymize users in published graphs. The active attack proceeds with the adversary inserting a small number of sybil nodes in the graph before it is released. The sybil nodes create links with a set of users whose privacy the adversary wishes to violate, a pattern of connections is also formed among the sybil nodes to make them conspicuous. Thus the adversary can efficiently find the sybils along with the targeted users in the published anonymized graph. The passive attack is more subtle, the members of the graph do not form new links but try to find themselves in the published graph. They then discover links among users to which they are connected. Both the suggested attacks proceed by searching for sub-graph

[14] `http://blog.kaggle.com/2011/01/14/how-i-did-it-will-cukierski-on-finishing-second-in-the-ijcnn-social-network-challenge/`

patterns in the released network and assume the published network to be unperturbed. While the active attack can be targeted it is likely to be detected using various sybil detection techniques [108–113]. Active attacks are hard to launch at scale; attacker cannot control the links made to the sybil nodes by the rest of the network, and many social networks only allow a link to be created if the connection is mutual. Passive attacks are also hard to launch at scale as colluding nodes can only breach the privacy of individuals they are already friends with. Additionally, both attacks are sensitive to perturbation in the released data.

Korula and Lattanzi [114] (`KL`) use seeds and common neighbors to reconcile users across social networks. Their algorithm is not meant to be a hostile attack but is aimed at aiding users with tasks such as suggesting friends. The algorithm starts with a high number of seeds, about 5-10%, and maps users across overlapping social networks starting from high degree nodes using common friends. Their technique is based on the absolute number of common friends and cannot attack low degree nodes. In summary, the adversarial model is unrealistic due to the high number of seeds needed to mount the attack.

Yartseva and Grossglauser [115] (`YG`) analyze the of success of seed-based social graph de-anonymization algorithms. They analyze the dependence of large scale de-anonymization on the number of seeds and propose ways to estimate the critical seed set size. A seed-based de-anonymization scheme similar to `KL`, based on common neighbors, is also proposed which is shown to depend upon a critical number of seeds, properties of the graph, the overlap with the auxiliary graph and common neighbor threshold for large scale percolation. The algorithm has a high error rate when vertex sets of auxiliary and sanitized graphs are not identical. Setting a threshold prevents it from attacking low degree nodes.

Ji *et al.* [116] quantify the de-anonymizability of social networks under seed knowledge. The authors investigate the feasibility of mounting seed-based de-anonymization attacks on social graphs from the theoretical perspective. A large scale evaluation is conducted using 24 real world social networks to demonstrate the conditions for perfect and imperfect de-anonymization and the number of users that can be perfectly de-anonymized. The results show that structural attacks are more potent than attacks based only on seed information. The analysis of structural attacks is limited as it does not leverage all the global graph properties for de-anonymization.

Ji *et al.* [103] (`JLS+`) present two seed-based attacks that use structural similarity, common neighbors and seed-induced distance between nodes to identify node mappings. The first attack needs the precise knowledge of overlap between the two graphs whereas the second attack estimates this overlap. The two-phase algorithm starts with seed selection and then propagates the seed mappings iteratively to neighboring nodes. Each iteration uses the already mapped nodes to expand the mappings. The technique is resistant to moderate noise levels and achieves good accuracy. The attack is based on graph metric heuristics with arbitrarily chosen weighing parameters. The auxiliary graph used to attack the real

graph are structurally very similar while some datasets are hand-curated. The techniques presented are similar to the attack presented by Narayanan and Shmatikov [5], although the authors conduct a broader study.

2.8.2 Seedless attacks

Ji *et al.* [104] (`JLSB`) study the de-anonymizability of social networks using structural information. They highlight the conditions under which perfect and imperfect de-anonymization can be achieved. Although, most social networks can be largely de-anonymized it is shown to be computationally infeasible to search for such an optimal attack. A computationally feasible optimization-based de-anonymization attack (which is a relaxed version of the optimum scheme) is proposed as a compromise. The algorithm assumes the auxiliary and sanitized graphs to have the same node set which is then used to start de-anonymization by mapping highest degree nodes from both the graphs. In absence of complete match the algorithm may suffer from high error rates or not even launch. Additionally, the algorithm does not consider mapping nodes in the case where their degrees have been disproportionately damaged thus skewing the correspondence between high degree nodes. Heuristics with arbitrarily chosen parameters are used to map nodes based on structural similarity by minimizing de-anonymization error. The results are good for optimization-based de-anonymization, though they are run on graphs with high overlap. The theoretical study sheds light on the difficulties in preserving privacy while publishing structural datasets.

Wondracek *et al.* [117] employ web-browser history-stealing attacks to demonstrate that group membership of users of a social network is sufficient to de-anonymize them. The attack has important privacy implications but requires active effort by the adversary.

Perdarsani *et al.* [118] (`PFG`) propose a Bayesian framework to match common nodes across social networks without seeds. The algorithm proceeds in rounds starting with mapping high degree nodes based on degree fingerprints. As most likely nodes are mapped they generate additional features based on distance to the mapped nodes which are used to map increasing number of nodes in subsequent rounds. The algorithm presented is limited in scope as the process relies on computing the probability of node pairs being identical or non-identical based on their features. This requires knowledge of the anonymization scheme to create a probabilistic model and even after this knowledge it limits the features used to represent nodes as the model complexity increases with that of the features. Additionally, the algorithm suffers from inefficiency with a time complexity of $\mathcal{O}(n^3 \log n)$ where n is the number of nodes in the graph. The proposed strategy is very sensitive to graph overlap and needs graphs to be sufficiently similar with a large node overlap. We present an improved de-anonymization algorithm in Chapter 5 which is free from such limitations as the model is learned by the classifier automatically in the training phase.

Perdarsani and Grossglauser [119] study the conditions under which the structural cor-

relation of two social graphs sampled with a noise from a larger graph becomes possible. They investigate the dependence of graph anonymity on its parameters irrespective of the graph de-anonymization algorithm applied. It is shown that the mean node degree needs to grow only slightly faster than $\log n$ for nodes of a random graph with n nodes to be identifiable. Although, the existence of an algorithm which achieves perfect matching has been demonstrated, it remains a challenge to discover it. The work provides further evidence regarding feasibility of graph de-anonymization using topology alone, and highlights the challenges involved in releasing such data.

2.9 Definitions

In this section we define terms that are commonly used throughout the subsequent chapters.

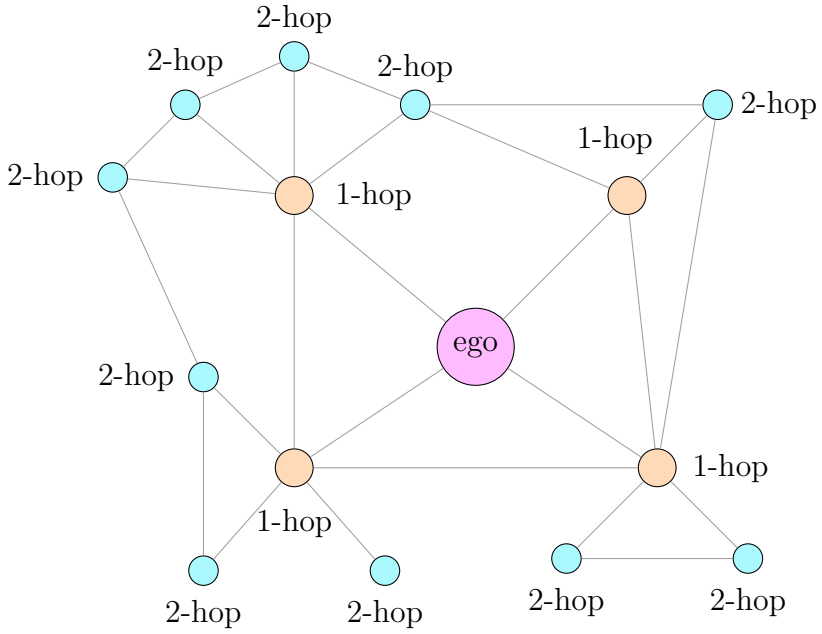


Figure 2.2: The 2-hop egonet of a node

Ego An individual focal node in a graph; a graph has as many egos as it has nodes.

Egonet The local network centered around an ego derived based on some function.

***i*-hop node** A node such that the shortest path length from the node to the ego is i.

***i*-hop network** A node induced neighborhood graph around an ego with all i-hop nodes included. It is also known as i-hop neighborhood.

Graph density The fraction of total possible edges that exist in a graph. It is formally defined as:

$$\frac{2 \cdot m}{n \cdot (n-1)}$$

where m is number of edges and n is the number of nodes in the graph.

Average node degree The average degree of a node in a graph, defined as:

$$\frac{2 \cdot m}{n}$$

where m is number of edges and n is the number of nodes.

Figure 2.2 shows the 2-hop egonet of a node formed around its 1-hop and 2-hop neighbors.

2.10 Summary

In this chapter we studied the privacy landscape and challenges concerning the release of highly-dimensional datasets. Starting with privacy in high-dimensional datasets we saw how it is related to social networks and why it is so hard to preserve when releasing datasets. We then took a detailed look at the privacy preserving approaches developed so far – ranging from interactive to non-interactive mechanisms. Most of the approaches so far have been developed as a response to the attacks on privacy which have always been a step ahead. As we saw a wide variety of attacks have been proposed to breach privacy in social networks – seed-based and seedless. So far the attackers seem to be winning. In the subsequent chapters we propose a new generation of social graph attacks by replacing the heuristic based algorithms with learning models and describe how it fits in the present context. Finally, we also discuss ways to mitigate risks associated to releasing social graph data by limiting its distribution and providing data analysis platforms while still controlling the datasets.

Chapter 3

Automating social graph de-anonymization

3.1 Introduction

A number of the serious privacy cases already discussed, such as the Netflix scandal, have shown that anonymization of social networks is much harder than it looks. Rich datasets have are often published for research purposes with only casual attempts to anonymize them. Research in de-anonymization has also seen an upswing [4, 87, 117, 120], leading to high-profile data releases being followed by high-profile privacy breaches. These developments have forced organizations to make some effort to better anonymize the released data. However, distorting data to achieve this contradicts the very purpose of a release, since it damages utility. So how hard can it be to re-identify users? In this chapter we present a generic and automated approach to re-identifying nodes in anonymized social networks which enables novel anonymization techniques to be quickly evaluated. It uses a machine-learning model to match pairs of nodes in disparate anonymized subgraphs.

Social network graphs in particular are high-dimensional and feature-rich data sets, and it is extremely hard to preserve their anonymity. Thus, any anonymization scheme has to be evaluated in detail, including those with a sound theoretical basis [61]. As discussed in §§ 2.4 to 2.6, many techniques have been proposed to resist de-anonymization; however Dwork and Naor have shown [121] that preserving privacy of an individual whose data is released cannot be achieved in general. The resulting uncertainty makes mass data release a very tricky proposition specially from the perspective of data subjects.

Ad-hoc vs generic. It has been conclusively demonstrated that merely removing identifiers in social network datasets is not sufficient to guarantee privacy. Despite these results, data practitioners, continue to propose anonymization strategies in the hope that they can resist de-anonymization "in practice", such as the ones used to protect datasets from

The Data for Development (D4D) challenge. This has led to a *cat-and-mouse game*: Data practitioners devise new anonymization variants by tweaking simple building blocks like sampling, deleting nodes or edges or injecting random ones. Then privacy researchers devise ad-hoc de-anonymization attacks to break the new variants.

Unraveling each anonymization technique manually, requires considerable effort and time and each attack can be defeated by a small tweak to the anonymization strategy, often by destroying specific features the attack exploited. Tailoring attacks to specific scenarios [3] highlights the problem of anonymization. However, it does little to deter future attempts to formulate "novel" anonymization techniques. Additionally, the expense involved in evaluating each new scheme cannot be amortized.

We need generic de-anonymization techniques that will allow cheap and timely evaluation of novel anonymization schemes, and eliminate large classes of weak ones. In this chapter, we demonstrate the efficacy of automated de-anonymization attacks on real-world anonymization schemes. They automatically uncover artefacts remaining after anonymization that enable re-identification of nodes in social networks. Our automated attacks can be used quickly and cheaply to screen "novel" anonymization schemes.

Our contributions. The key contribution of this work is to cast the problem of de-anonymization in social networks as a learning problem, and show that an automated learning algorithm can be used to evaluate a variety of social network anonymization strategies. Specifically, we:

- Formulate the problem of de-anonymization in social networks as a learning task. From a set of examples of known correspondences between nodes (*training* data) we wish to learn a good de-anonymization model (§ 3.3.1).

- Describe a non-parametric learning algorithm tailored to the de-anonymization learning problem in social graphs. The algorithm is based on random decision forests, with custom features that match social network nodes and a granular graph structure-based metric to capture the likelihood of node re-identification (§§ 3.3.2 to 3.3.4).

- Evaluate the learning algorithm on a real-world de-anonymization task from the D4D challenge, and compare it with an ad-hoc approach (§ 3.4.4).

- Show that the algorithm and model learn sufficient information about the anonymization algorithm, rather than the specific dataset anonymized, to be useful when de-anonymizing social networks of a different nature than the ones used for training (§ 3.4.3).

- We apply the automated learning algorithm, to a standard problem [5] of de-anonymizing nodes across social networks. It performs well, even when a very small number of examples are used to train it (§§ 3.4.4 and 3.4.5).

The work presented in this chapter is based on the paper titled *An Automated Social Graph De-anonymization Technique* [122] published in the Proceedings of the 13th Workshop on Privacy in the Electronic Society (WPES 2014), and is in collaboration with George Danezis. I did all the day-to-day work while George acted as a senior academic advisor for this work. George proposed the idea of using machine learning to de-anonymize social networks. The design of the system evolved through our discussions. I wrote the code and conducted all the experiments. The first draft of the paper was written by me and was subsequently reviewed by George.

3.2 Motivation: the D4D challenge

In July 2012 Orange unveiled the *Data for Development* (D4D) challenge[1] to promote research in behavioral science in the Ivory Coast using a number of mobile call datasets. The datasets are based on "anonymized" *call detail records* extracted from Orange's customer base. The data was collected for 150 days, from December 1, 2011 until April 28, 2012, and contains 2.5 Billion calls and SMS exchanges between around 5 Million users, i.e. a quarter of the Ivory Coast population [123]. Teams with access to the datasets had to sign appropriate non-disclosure and ethics agreements to protect privacy and supplement the anonymization procedures employed.

Prior to release, the organizers asked us to evaluate the quality of the anonymization scheme used. We examined the datasets, proposed an ad-hoc de-anonymization mechanism, and advised them accordingly. As a result the organizers independently modified the anonymization process (Scheme 2) prior to the final release. In total four datasets were released for analysis [124]. In this work we concentrate on analyzing the anonymization procedures for *dataset 4*, representing an anonymized phone call graph. We study two versions of the anonymization scheme that were used for dataset 4: the first is the scheme (Scheme 1) we were provided for evaluation, and the second (Scheme 2) is the scheme they devised for the final data release. We compare the two strategies, analyze them and present de-anonymization attacks for both of them. We note that none of our results are based on processing the actual D4D datasets – we evaluate the anonymization process only, using datasets with known ground truth.

[1] `http://www.d4d.orange.com/`

3.2.1 Data release and anonymization

The two variants of the anonymization process select a number of highly connected individuals (egos) and output a social subgraph around them (egonets).

Scheme 1. The initial dataset contains subgraphs of about 8 300 randomly selected individuals. Each subgraph contains all communication amongst the egos and their contacts for up to two degrees of separation. The data also includes the volume of calls, duration of calls within a period and their directionality. The data spans a period of 150 days which is split into two-week chunks. Individuals are assigned random identifiers which remain the same across time slots but are unique to each subgraph. This is meant to obfuscate links between subgraphs to prevent an adversary from reconstructing a larger communication graph by stitching subgraphs together. We refer to this anonymization strategy as "Scheme 1".

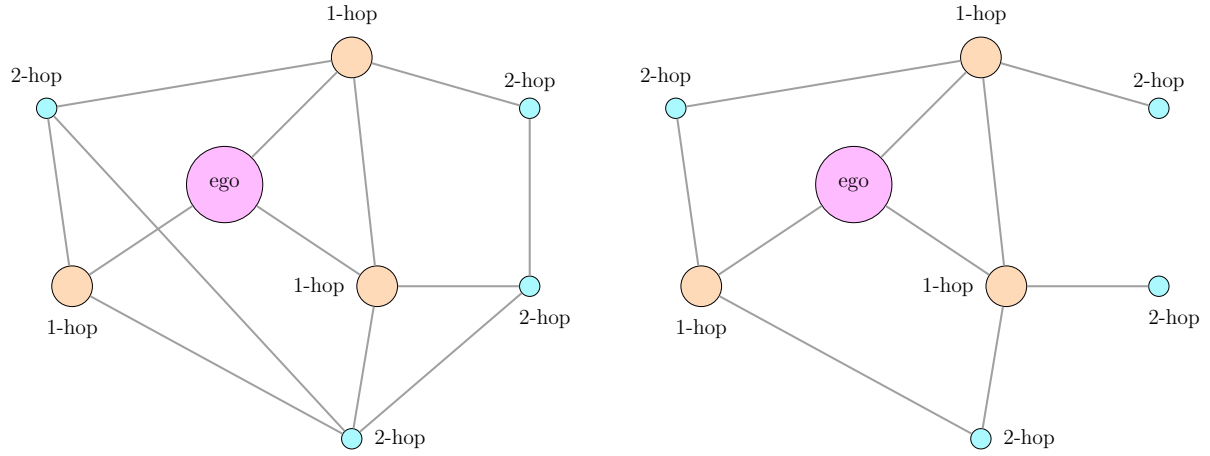
It is extremely hard to anonymize graph data without preventing meaningful analysis of the data, the methods used to analyze graphs can equivalently be used to compromise privacy, our analyses of the datasets corroborates this theory. Difficulty in anonymization is specially apparent for dataset 4 which releases high-dimensional and feature-rich data like graph topology, research suggests that such data is almost impossible to anonymize efficiently without seriously limiting its usage [3, 6, 125].

Scheme 2. Following our preliminary analysis the organizers independently modified the anonymization scheme prior to the final release:

1. the number of egonets released was reduced from 8 300 to 5 000;
2. all links between the 2-hop nodes (friend-of-friend to friend-of-friend of the ego) were deleted;
3. the number of calls, duration of calls and the directionality of calls between two users was removed;
4. calls with easy to re-identify features were removed, such as to or from public phones.

These changes degrade the information available to adversaries but, as we demonstrate, they do not guarantee anonymity. We refer to the modified anonymization strategy as "Scheme 2".

The toy example presented in Figures 3.1a and 3.1b illustrates the difference between the two schemes when anonymizing the same ego subgraph. The nodes represent individuals and the edges indicate their communications. The entire 2-hop ego network has been released for Scheme 1, whereas the interactions between the 2-hop nodes have been deleted in Scheme 2.



(a) Scheme 1 (b) Scheme 2

Figure 3.1: Scheme 1 preserves the complete 2-hop network while Scheme 2 removes edges between 2-hop nodes

3.2.2 Robustness of anonymization

The intent behind both anonymization schemes is to obscure the full social graph by only releasing unlinkable subgraphs. Releasing the full graph would lead to de-anonymization risks similar to those against Netflix [3] or AOL[2] (see, § 2.1.1). Therefore the primary aim of our privacy evaluation is to determine the extent to which the same node in different anonymized sub-nets may or may not be linkable. Special attention needs to be paid to the *false positive rate*, namely when we label two nodes as identical when in fact they are not. Even a small false positive rate may in general introduce considerable noise in piecing together subgraphs, since matching a node to a n-node subgraph may only contain a single match but at least $n - 1$ mismatches.

Success of de-anonymization. The success of de-anonymization is measured by an individual's risk of re-identification [3, 5, 117]. To model this we measure the probability of re-linking an individual present in two social graphs purely based on topology. More formally, we measure the indistinguishability of an identical pair of individuals from a random pair. Members of the pair belong to different social networks under evaluation.

We note that we have concentrated on matching nodes between egonets as a measure of success of de-anonymization. This is the task we were required to perform when evaluating the actual D4D dataset on behalf of the competition organizers. However, this reduces to a lower bound on the probability of matching an anonymized egonet with a fuller social graph or a fragment of the social graph. One may simply take the full social graph and apply the anonymization procedure to generate egonets. Then our techniques can be

[2] `http://www.nytimes.com/2006/08/09/technology/09aol.html`

applied to determine a match. However, we also discuss how to apply our technique directly.

Finally, our success rates need to be interpreted as lower bounds. We use graph-metric based node features that are not complete: better ones may be found that improve the de-anonymization rate. This is particularly true of the node features used as part of our machine learning approach. Automatically trained classifiers are capable of out-performing humans in many tasks, but are limited when it comes to using higher level features. Thus, even when de-anonymization rates are low, our results can never be interpreted as a proof of security, only an illustration of vulnerability.

Graph isomorphism problem. The challenge of matching social graphs to reconcile identities of individuals is rooted in the *graph isomorphism problem* [126]. Two graphs G and H are considered isomorphic if there is a bijective mapping between the node sets of the graphs such that it preserves the edges between the nodes in both graphs. Babai and Luks [127] proposed a solution to the problem with a runtime complexity of $2^{\mathcal{O}(\sqrt{n \log n})}$, this was subsequently improved by Babai [128] to $2^{\mathcal{O}(\log^c n)}$; where n is the number of nodes in the graph. Czajka and Pandurangan [129] show that efficient solutions for graph isomorphism problem exist for random graphs. Some variants of the problem are *subgraph isomorphism problem* [130, 131] – determining if G contains a subgraph that is isomorphic to H and *maximum common subgraph isomorphism problem* [132, 133] – finding the largest subgraph of G that is isomorphic to a subgraph of H. No polynomial time algorithms are known for these problems in the general setting. The problem of mapping nodes across overlapping social graphs is even harder due to the noisy and dynamic nature of the graphs and relations between the nodes. As seen in Chapter 2 so far the approach to solve these problems has been to recover an imperfect matching often using seed mappings to initiate the process.

3.2.3 Ad-hoc de-anonymization

We first present an ad-hoc attack on Scheme 1 that links nodes across egonets. We observe that we may encounter three distinct cases when linking nodes from two different egonets, that allow different types of analysis:

Case 1: "1-hop". Both nodes are at a distance of 1-hop from the ego, or are the ego in both subgraphs;

Case 2: "1,2-hop". Only one of the two nodes is 2-hops from the ego in one subgraph;

Case 3: "2-hop". Both nodes are at a distance of 2-hops from the ego in both subgraphs.

Given a large egonet it is easy to detect which case we are tackling by first identifying the ego. In a medium-sized subgraph it is the single node from which all others are at most 2-hops away. If more than one node satisfies this relation we cannot detect the ego accurately. However, this does not affect the attack.

The ad-hoc attack exploits the variability of degree distributions of nodes in social networks, which is known to approximately follow a power-law [134]. We observe that the entire 1-hop neighborhood of nodes that are friends of the ego is preserved in Scheme 1 (since the full degree 2 graph is extracted). Therefore for Case 1 node pairs we can generate a *signature* for nodes that is invariant under the anonymization procedure: the signature for each node consists of the sorted list of degrees of all the nodes in its *1-hop* network. For example, if a 1-hop node has four friends with degree 3, 2, 2 and 1 in its 1-hop network then its signature is {1, 2, 2, 3, 4}. When a large signature of two 1-hop nodes, in different egonets, matches exactly we classify them as being the same node. We present an evaluation of the effectiveness of this attack in § 3.4.

3.2.4 Limitations of ad-hoc de-anonymization

We tried to extend the attack to node pairs in Case 2 and Case 3 by using an approximate match on their common 1-hop nodes. Since considerable parts of the neighborhood of 2-hop nodes are deleted we cannot use an exact signature match on them. Perhaps unsurprisingly, we got mixed results for this approach and they were neither consistent nor robust. Extending the attack to Scheme 2 was even more restrictive; the exact signature match is not applicable for Case 1 due to the edges being deleted in relation to the ego, thus precluding any extension to other categories.

The failure to generalize this approach is quite limiting: overlaps between egonets are likely to contain a 2-hop node rather than exclusively 1-hop nodes, since the number of 2-hop nodes is considerably larger. The attack on node pairs involving 2-hop nodes cannot be mounted unless the egonet pair has some 1-hop nodes in common. Due to the unsatisfactory results we do not discuss an ad-hoc extension of this attack any further, and simply consider it inapplicable for Scheme 1 Cases 2 and 3 and Scheme 2 completely.

Overall, constructing such ad-hoc attacks is an expensive procedure. It requires *manual* identification of some *invariant property* between subgraphs. It is also rare that such ad-hoc attacks make use of a combination of weak features – since identifying them by hand would be a laborious process. A reliance on quasi-invariant features makes such ad-hoc attacks quite fragile. This is exemplified by the anonymization procedure of Scheme 2 that severely degrades features vital to the ad-hoc attack, namely the edges between the 2-hop nodes which generates inconsistent signatures of 1-hop nodes and renders our attack completely useless.

This points to a more significant problem, relating to the *economics of privacy research*:

designing new variants of anonymization procedures is cheap; analyzing them requires manual labor to extract new complex invariants, even when the schemes are quite obviously leaking a lot of identifying information. Attacking even simple anonymization techniques individually is not practical and provides diminishing returns. Instead, we propose a general approach that applies to the analysis of anonymized social graphs. It uses machine learning to automatically learn invariants or informative relations between the same node in different egonets or subgraphs.

3.3 Learning de-anonymization

Graphs carry a huge amount of information and research has shown that it is notoriously hard to release graph data for meaningful analysis while preserving privacy. Although successful and efficient attacks have been presented, they are hand-crafted for a particular problem. As illustrated by our analysis of Scheme 1 and the slight modifications in Scheme 2 this approach gets expensive for the analyst. To tackle this challenge we use techniques from machine learning to automatically uncover and learn invariants and similarities that can be used to link data in anonymized graphs. We present a classifier using random forests, and use it to mount attacks on Scheme 1 and Scheme 2 using a common algorithm despite their differences. The results are presented in § 3.4.

3.3.1 De-anonymization: a learning problem

A large number of variants of anonymization algorithms can be devised at low cost through combinations of sampling, injecting random nodes, removing specific or random nodes, or picking specific subgraph. Each variant would traditionally require careful manual analysis to devise an effective de-anonymization strategy. Thus the evaluation of such algorithms – and the demonstration that they perform poorly – is a labor intensive process. However we observe a key commonality of "useful" anonymization procedures: they need to *preserve a rich set of generic features of the original graph, and its subgraphs, to provide an acceptable degree of utility to legitimate data users.* As a consequence a subset of those features may be automatically refined as identifiers to de-anonymize the nodes.

We propose to replace the manual process of devising de-anonymization strategies by a generic learning algorithm. The learning algorithm takes an anonymization algorithm and automatically learns a model that allows the de-anonymization of a significant fraction of social network nodes. The learning algorithm does not require a full description of the anonymization scheme. Instead, we provide the learning algorithm with examples of network nodes that are meant to be unlinkable. The algorithm then automatically learns sets of features that constitute effective invariants that allow linking and de-anonymization.

Two de-anonymization problems are considered in this chapter within the generic learning framework:

- **Linkage between sub-nets:** Two nodes in two different anonymized subgraphs are presented to the algorithm, which must decide whether they represent the same subject or not. This is the problem associated with linking ego-nets in the D4D challenge. This task forms the core of our evaluation.
- **Linkage between raw and anonymized graphs:** A vertex in a raw graph and a vertex in an anonymized graph are presented, and the algorithm has to decide whether they represent the same user. This more traditional task is discussed in § 3.4.4.

The models learned for the tasks in both cases above are presented with two target user nodes in two different subgraphs. Ultimately, their task is to classify the two target graph nodes as representing the same user or not.

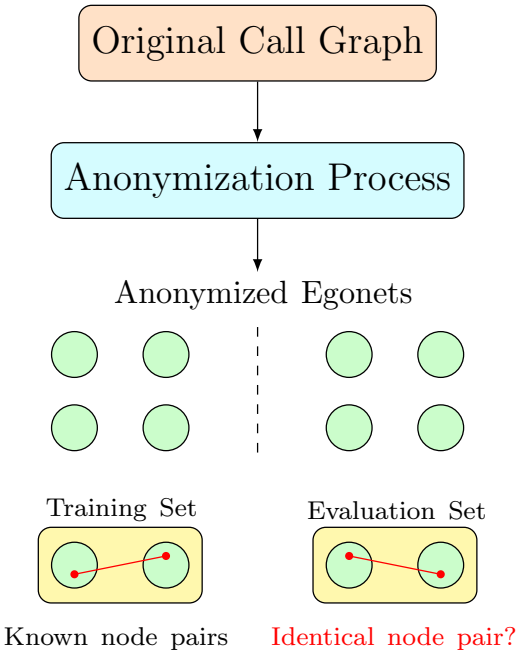


Figure 3.2: The model for D4D sub-net linking learning task

The role of training examples. Our learning algorithm makes use of examples of anonymized graphs, to learn invariants that can be used to link nodes. These can be generated at will by applying a black-box anonymization algorithm. It is best if the training examples come from the same distribution as the target anonymized networks. For example, the D4D challenge publishes mobile phone call graphs, so using some mobile phone call graph to generate training examples is best. Finding *some* example call records, and processing them to produce examples of anonymized subgraphs is easy for a serious

adversary. We note that the training examples do not have to contain any nodes that will be in the actual target anonymized graphs, unlike previous approaches [5] that required known *seeds* to bridge anonymized and raw graphs. Thus, for example, one may use an available call subgraph from one operator to train an attack on anonymized call graph from a competing operator. In § 3.4.2 we provide an extensive analysis in this setting, using very few training examples.

However, we note that training examples uncover artefacts of the anonymization technique that enable attacks, not merely quirks of the specific training data. This is illustrated in § 3.4.3 by the fact that training examples from graphs of a very different nature, leads to de-anonymization success in other graphs. Figure 3.2 illustrates the sub-net linking learning task; the model learns to de-anonymize nodes by training based on a set of egonets that are different from those being attacked.

3.3.2 Decision trees and random forests

Decision trees and random forests have been widely used in computer vision with significant success and better results than other machine learning techniques [135–138]; *Microsoft Kinect for XBox 360* being one of the most recent achievements [139]. Our generic model for de-anonymization is represented as a collection (forest) of decision trees on graph features, which are learned using a random decision forest training algorithm.

A random decision forest is an ensemble of randomly trained decision trees [140, 141] and was first introduced in the context of hand-written digit recognition [141, 142]. However, ensemble methods were proposed even earlier by Schapire [143] as a boosting algorithm which builds accurate *strong* classifiers as a linear combination of many weak ones by iteratively re-weighting training data. Each tree of a random forest is a simple learner which provides a rough prediction. The prediction of trees, representing multiple learners of different types, are collated together resulting in a performance improvement [144–147]. A detailed treatment of decision forests and their applications can be found in the technical report by Criminisi *et al.* [148]. Here we present a brief overview of the techniques.

Decision trees. A decision tree is comprised of nodes and edges arranged in a hierarchical structure. We use binary trees in our work because using n-ary trees ($n > 2$) does not provide significant accuracy benefits [148]. The branch nodes of the tree are called *split nodes* and the terminal nodes are the *leaf nodes*. A decision tree uses a weak classifier at each split node to route an item to the "left" or "right" child node based on its features. The item is routed down the tree, ultimately reaching a leaf node where one class, or a distribution over classes, is assigned. The features on which the split node predicates are defined is problem-specific – we propose features to represent an anonymized graph in § 3.3.3.

Training and testing. The tree-training phase involves injecting labeled data into the tree to optimize the split parameters at the internal nodes and determining the leaf predictors. Given a set of pre-labeled data points, each internal node finds a split of points based on some feature that produces maximum information gain as defined by Equation (3.1). Each internal node of the tree is associated with a binary classifier also known as *weak learner*, whose output decides the data split for a given parameter. Here $\mathcal{S}^L$ denotes the set of points passed to the left child of the split node, $\mathcal{S}^R$ denotes the set of points passed to the right child. We denote $\mathcal{S}$ as the set $\mathcal{S}^L \cup \mathcal{S}^R$ of all items used to train a split node. To train each split node we select the feature of items in $\mathcal{S}$ producing sub-sets $\mathcal{S}^L$ and $\mathcal{S}^R$ such that it maximizes the information gain objective function:

$$I = H(\mathcal{S}) - \sum_{i \in \{L,R\}} \frac{\mid \mathcal{S}^i \mid}{\mid \mathcal{S} \mid} H(\mathcal{S}^i) \tag{3.1}$$

where $H(\mathcal{S})$ is the Shannon entropy of the labels in set $\mathcal{S}$ with elements belonging to the set of classes $\mathcal{C}$ is given by:

$$H(\mathcal{S}) = -\sum_{c \in \mathcal{C}} \Pr[c] \log \Pr[c].$$

Maximizing the information gain at every split node decreases entropy of data and increases confidence in tree prediction. Each leaf node stores the empirical distribution of the classes associated with the subset of training data reaching that leaf node. The predictor is defined as the probability of a data point belonging to a particular class based on the distribution. We only use binary classification, hence we have just two classes.

The decision forest model. An ensemble of randomly trained decision trees constitutes a random decision forest. Randomness is injected into trees during training by:

- *Bagging:* a random subset of training data is used for each tree [142];
- *Randomized node optimization:* each split node is trained using a random subset of features [149].

These techniques are orthogonal and we use both of them to produce different trees as discussed in § 3.3.4.

The trees of a random forest are trained independently in parallel. A data point is classified by pushing it through all the trees, branching left or right within each tree according to the item features, until it reaches the leaves of all trees (this process can also be parallelized). The prediction of each tree at the corresponding leaf node is averaged to generate an aggregate forest prediction, defined as:

$$p(c|v) = \frac{1}{T} \sum_{t=1}^{T} p_t(c|v) \tag{3.2}$$

where $p(c|v)$ is the empirical probability (as computed by the random forest) that given a feature vector v, it belongs to a data point of class c, T is the number of trees and $p_t(c|v)$ is the prediction of an individual tree. We use $T = 400$ trees in our experiments.

3.3.3 Specialized social graph features

Random forests rely on finding features that when combined using many weak learners, classify the data accurately. Training ensures that the best features are retained, and their best conjunction will be selected per tree. Different trees allow disjunctions of features to contribute to classification.

A balance must be struck between too generic or too specific features. If the features are too generic, in that they appear in most subgraphs, then classification will be akin to guessing. On the other hand if they are too specific and rely on intricate graph structures, then they may be seriously damaged during anonymization. In all cases efficient feature extraction is important to allow for fast training and evaluation.

Node similarity metrics have been well known to significantly improve the confidence and accuracy in predicting links between nodes [86, 150–157]. Cukierski *et al.* [158] summarize techniques to differentiate between fake and real edges in a graph using similar features. The authors use a variety of features and random forests to reach their final result. Although a much simpler problem compared to link prediction, the features help in finding similarity scores between nodes.

Henderson *et al.* [159] define regional features to represent a node in a graph. The features are extracted using node degree and egonet structure around the node. The experiments show the efficacy of these features in transferring learning from one graph to another by training a model using topological features to describe a node. The features are scalable, robust to noise and perform well in general data mining tasks such as node classification and social graph de-anonymization. Nunes *et al.* [160] propose techniques to resolve user identifiers in social networks using binary classification. The task is similar to ours and a classification model is used to distinguish between a pair of identifiers as belonging to the same individual or not. The features used to describe a pair of identifiers consist of similarity between profile information, interests and friend lists. Results indicate that the user identities can be distinguished with a high success rate.

Inspired by the results from the confluence of link prediction and machine learning literature we present a solution which uses these techniques for social graph de-anonymization. We use the degrees (number of friends) of the "friends" of a node as features. Degree distribution is a fundamental property of social networks [134] and a node can be uniquely identified by its neighborhood degree distribution [83]. Most anonymization strategies strive to preserve some utility of the data, and damaging such a fundamental property of graph elements *unpredictably* has an averse effect on utility. Hence, the very fact

node degrees are *predictably* perturbed also allows us to mount an attack: we expect the distribution of degrees in the neighborhood of subgraph nodes to be related between different ego-nets. As demonstrated in Chapter 4 too much damage to degree distribution renders the data useless. Furthermore, neighborhood degree distribution can be efficiently computed. These are generic graph features, and not specific to any anonymization scheme.

For each node in the social network we define a feature vector $v = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{Z}$ of length n made up of components which are bins of size b. Each component represents the number of neighbors that have a degree in a particular range. The i^{th} component is the count c_i of the number of neighbors with degree such that $i \cdot b <$ degree $\leq (i+1) \cdot b$, where $i \in \{0, n-1\}$. If the degree exceeds the maximum possible range of $n \cdot b$ then it is included in the last bin.

Neighborhood degrees are binned for efficiency and performance. Anonymization perturbs the degrees of nodes, but this change cannot vary vastly across subnets for the same node as this would damage data utility. By binning the degrees in a particular range we get a coarse estimate that is robust to perturbations due to anonymization. Since the exact degree of the nodes across different anonymized subgraphs may vary, the exact degree is irrelevant, and binning increases matching robustness. In our experiments we use $n = 70$ and $b = 15$. Figure 3.3 illustrates the feature vector $v = (8, 4, 0, 0, 3, 0, \ldots, 0, 2)$ of a node with neighbors of degrees – $\{1, 1, 3, 3, 5, 6, 7, 13, 16, 20, 21, 30, 65, 69, 72, 1030, 1100\}$. We illustrate the classification performance trade-off for different value of n and b in § 3.5.3.

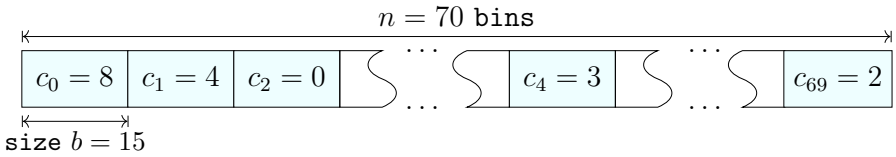


Figure 3.3: Example node feature vector

Apart from the depicted values all bins contain 0's. The feature vector represents a node in a particular egonet. Defining features in this manner allows us to compute them only for node pairs of interest and proceed with classification. Note that when a node appears in different egonets we expect the feature vectors to be different. However, through the training phase, we learn the regularities that are able to predict a match despite the differences.

3.3.4 Training and classification of node pairs

The basic data point to be classified consists of a pair of node features, each in a different egonet subgraph. Let us denote the feature vector belonging to a node n_p to be v_p. Each node pair (n_p, n_q) is represented as a pair of vectors (v_p, v_q). The node pair can be labeled as either "identical", if the nodes n_p and n_q are the same individual in different egonets;

or "non-identical" if n_p and n_q are in fact different individuals. The objective of the classification task is to infer the label of the pair (n_p, n_q) on the basis of the feature vectors v_p and v_q. If we succeed in classifying node pairs accurately then we may relabel identical node pairs across egonets to recover a larger graph.

Training & Weak Learner. We use both bagging [142] and randomized node optimization [149] to train decision trees. The trees are trained by injecting a random sample of data points at the root node. The sample contains data points of each class in equal proportion.

The weak learner at each split node is presented with pairs of feature vectors (v_p, v_q) and needs to decide whether they are assigned to the left or the right child node. To decide the split our weak learner uses the Silhouette Coefficient between two sample features $x \in v_p$ and $y \in v_q$ defined as:

$$\delta(x, y) = \begin{cases} 0 & \text{if } x = y = 0 \\ \dfrac{\mid x - y \mid}{\max(x, y)} & \text{otherwise} \end{cases} \tag{3.3}$$

where $x, y \in \mathbb{Z}$. Thus, for each feature pair (v_p, v_q) we can calculate all n^2 component pairs $\delta(v_p[i], v_q[j])$, where $i, j \in \{0, n-1\}$. Each feature vector has n components hence a pair of vectors have n^2 component pairs which can be inputs to the weak learner to split data. For a given component pair $(v_p[i], v_q[j])$ the tree computes $\delta(v_p[i], v_q[j])$ and passes the data point to the left or the right child depending upon threshold τ. As defined by Equation (3.3), $0 \leq \delta(x, y) \leq 1$; during training each split node is assigned a τ that splits the data for a given $(v_p[i], v_q[j])$ to maximize information gain. To choose the best τ for a given $(v_p[i], v_q[j])$ we cycle through $\tau \in [0, 1]$ in steps of 0.05. We inject randomness in the training of each split node by considering only a random 5% of the total n^2 $(v_p[i], v_q[j])$ tuples of features.

Once the values (i, j, τ) that best minimize entropy are determined for the split node, they are stored and do not change. The training procedure is repeated for its child nodes. We stop growing trees when the number of data points reaching a split node falls below 10% of the total data points that were injected at the root node. This provides a good balance between trees that are too deep or too shallow, both of which provide poor results. Training ensures that the most informative features out of those available are learned at each split node. Randomizing the available set of features produces robust forests, that classify data according to a diverse set of mutually supporting features.

Figure 3.4 illustrates a sample decision tree: the split nodes contain the weak learner parameters, the left branches correspond to a *False* result while the right branches correspond to *True*. The leaf nodes store the count of (non-identical, identical) vector pairs reaching that leaf. We note that the sets of features selected are, perhaps surprisingly, not

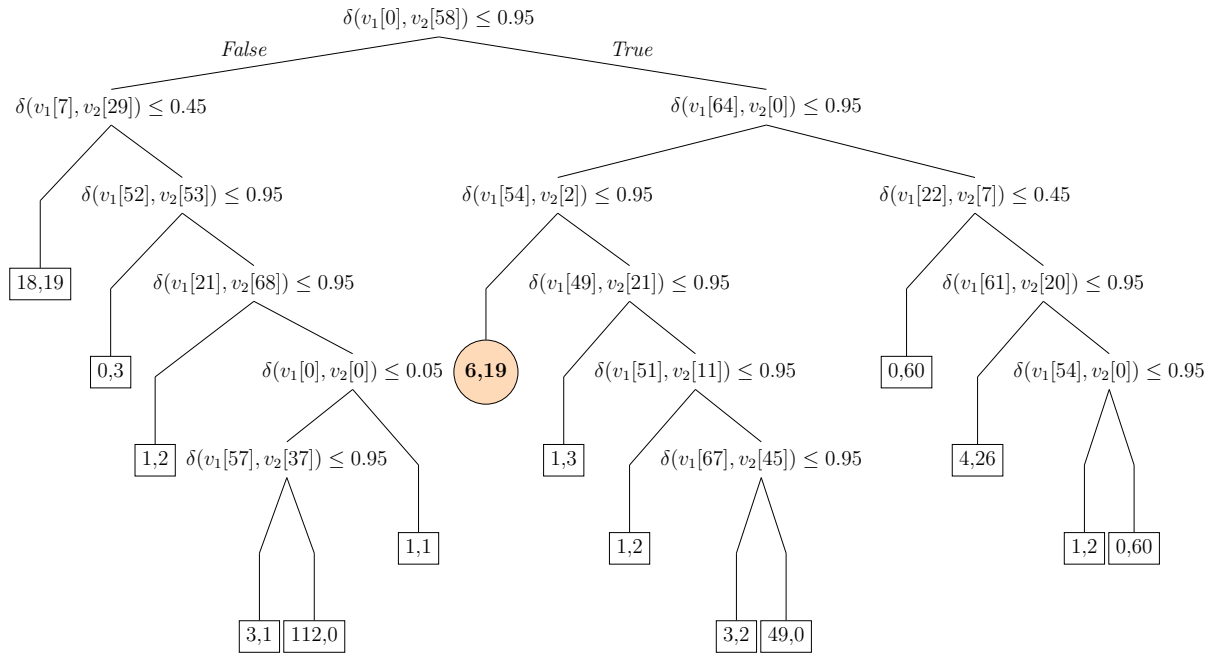


Figure 3.4: A randomly trained decision tree. The split nodes store weak learners $\delta(v_p[i], v_q[j]) \leq \tau$, the highlighted leaf node has posterior of $(0.24, 0.76)$

always from buckets close to each other. This is counterintuitive but works, illustrating the difficulty of choosing such features manually.

Classification. This is the simplest part of the algorithm. Once the decision forest has been trained, a previously unseen data point is classified by each tree till it reaches the leaf node and the values at all leaf nodes are recorded. At each leaf we calculate its probability of corresponding to an identical or non-identical node pair as their empirical distribution. For the highlighted leaf node in Figure 3.4 the empirical distribution is $(6, 19)$ and the probability of being classified as non-identical $= \frac{6}{6+19} = 0.24$ and probability of being classified as identical $= \frac{19}{6+19} = 0.76$. After traversing each tree in the forest all probabilities are averaged, as shown in Equation (3.2), to compute the final prediction. We discuss our choices of random forest algorithm parameters in § 3.5.3.

3.4 Evaluation

We evaluate the classifier by training and testing on publicly available real world social networks. Our analysis is based on datasets obtained from the *Stanford Network Analysis Platform*[3]. We use two social networks to demonstrate our results: Epinions – an online social trust network of a consumer review site and Pokec – the most popular online social

[3] `https://snap.stanford.edu/data/index.html`

network in Slovakia. These different types of network illustrate the applicability of our automated de-anonymization techniques across different types of social graphs.

We use a very small partition of each of those networks (see, § 3.4.1) to produce labeled training sets for the two anonymization schemes under consideration. Random forests are trained on the labeled training sets, and evaluated on separate test sets. Successful classification of test node pairs as "identical" or "non-identical" illustrates the success an adversary would have in using our techniques to attack the anonymization schemes. Social networks have similar properties, the decision trees aim to learn the features which distinguish a node based on these properties. The features may vary depending upon the anonymization strategy employed. Training the classifier on pre-labeled node pairs allows us to de-anonymize previously unseen node pairs.

To fully characterize the difficulty of de-anonymizing each type of node pair (1-hop, 1-to-2-hop or 2-hop, see § 3.2.3) we also consider each category separately. In those cases we set aside training and test data containing only pairs for that category. This does not provide the adversary with any advantage, since the category of a node and node pair can be inferred easily from the anonymized egonets. We also provide an aggregate analysis with node pairs from mixed categories. This is meant as a baseline representing the most generic attack, where the adversary is not making use of category information.

3.4.1 Experimental setup

The D4D datasets motivating this study comprised of 5 000 egonets of nodes selected at random out of customer base of about 5 Million. In line with this we generate 100 egonets for the Epinions dataset which has 75 879 nodes and 1 000 egonets for the Pokec dataset which has 1 632 803 nodes. We select those 2-hop egonets to have more than 400 nodes for Epinions and 800 for Pokec (to focus on "important" nodes as in previous work). The reason for doing this is two-fold. Firstly, since the egonets were released for analysis, too few members would make the dataset useless. Secondly, we found that selecting egonets of size below 400 did not produce sufficient common node pairs to run our analyses. The Case 1 node pairs are the rarest, we found less than 1000 of them for Epinions dataset (100 egonets and 4950 egonet pairs). The Pokec network is even more sparse, we found negligible Case 1 node pairs when using Scheme 2, hence to facilitate analyses we increased the minimum size of egonets to 800. Considering that Pokec has over 1.6 Million nodes, this change is minuscule. We only study linkage of nodes with degree greater than five; lower degree nodes have too little information to be meaningfully re-identified in bulk.

Table 3.8 in § 3.4.6 lists the number of training links extracted from the training ego-nets for different node pair categories. We make no distinction between categories of non-identical node pairs. For each classifier we train 400 trees, each with 600 identical node pairs (of the appropriate category) and 600 non-identical node pairs (1200 in total).

Figures of merit. The output of a trained random forest classifier is a real value in $[0, 1]$, where a higher score denotes a higher probability of nodes being non-identical. Decisions based on different thresholds lead to a trade-off between *True Positives* (TP) (the fraction of identical nodes that are labeled as identical), and *False Positives* (FP) (the fraction of non-identical nodes that are erroneously classified as identical). For each classifier, each dataset and each scheme we present the true positive rate, for selected values of false positives rates (0.01%, 0.1%, 1% 10%, and 25%). The Receiver Operating Characteristic (ROC) curves illustrate all trade-off points between TP and FP. A summary of the quality of the classifier is provided by the "area under the curve" (AUC) of the ROC curve. The diagonal line denotes the FP vs TP akin to guessing and produces an AUC of 0.5.

3.4.2 Results: same training distribution

We first present the results of classification of identical versus non-identical nodes when the training data comes from the same distribution as the data to be classified.

Epinions. Table 3.1, summarizes the performance of the *ad-hoc attack* in re-identifying node pairs of Case 1, Scheme 1. This hand crafted specialized attack leads to almost perfect results. On 900 identical node pairs it identified each one of them; on testing for 4907 non-identical node pairs the algorithm was correct 99.98% of the time. This is almost perfect classification, but the technique cannot be generalized to other cases or schemes.

Table 3.1: Success percentage for ad-hoc de-anonymization of Scheme 1 for Case 1 node pairs

	Success percentage	
	Identical	Non-identical
Epinions	100	99.98
Pokec	100	99.96

The automatic machine learning approach can be generalized to all cases of both schemes, as summarized in Table 3.2. For the sake of direct comparison, for Scheme 1 using a fixed small false positive rate (0.1%) we can recover 90.39% of Case 1 links; we also recover 46.82% of Case 2 links, and 17.36% of Case 3 links. The attack generalizes to Scheme 2 where we can recover 52.92%, 25.95% and 7.33% of links in Cases 1, 2 and 3 respectively. Despite Scheme 2 leading to lower re-identification rates, for a fixed false positive rate, we observe they are never negligible, and a released dataset is likely to lead to the linkage of a significant number of individuals between "unlikable" subgraphs. For even smaller false positives a significant number of Case 1 links are recoverable: 70.81% for Scheme 1 and 35.08% for Scheme 2. Case 2 linkage is also never negligible and higher than 11.37% for

both Schemes. If the adversary has any side information (such as seed nodes) that can be used to establish a good prior, a higher rate of false positives (1%) can be tolerated with linkage rates of over 35.67% even for the hardest Case 3 links.

Figure 3.5 illustrates the ROC curves for Scheme 1 and Scheme 2 for all types of node pairs along with area under the ROC curve (AUC). We note that the AUC for Scheme 2 is consistently above 95% for all link categories, higher than for Scheme 1 for Cases 2 and 3. This is due to the classifier performing significantly better for Scheme 2 when high false positive rates can be tolerated – which may or may not be the case depending on adversary side information.

Overall, the machine learning approach yields good results for Scheme 1, Cases 2 and 3 which the ad-hoc de-anonymization could not attack. More importantly, it yields good results for Scheme 2, on which the ad-hoc attack was not applicable. As expected, we get better results for the 1-hop category of node pairs, and for Scheme 1, which retain the most information from the original graph.

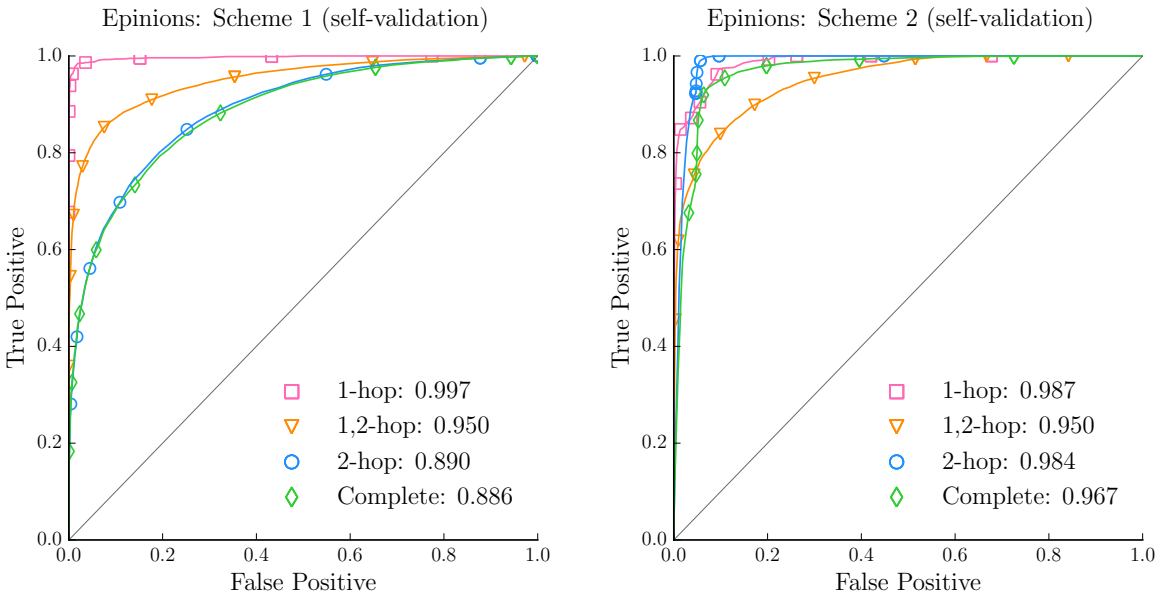


Figure 3.5: Epinions (self-validation): ROC curves for both schemes

Pokec. The results on the Pokec network are summarized in Table 3.3 while Figure 3.6 illustrates the ROC curve and the AUC. The results are similar to Epinions, which supports the general applicability of the attack. The ad-hoc attack against Scheme 1 gives almost perfect results. For Case 1 and 2 links the true positive linkage rate of the machine learning re-identification procedure is 42.92% and 11.58% for Scheme 1; for Scheme 2 they are 16.26% and 6.41% respectively (for a 0.1% false positive rate). Interestingly, the linkage rate for Case 3, which was the lowest in the Epinions network, now outperforms Case 2 and is on par with Case 1 when higher false positive rates are tolerable. This is likely due to a lower degree of overlap in the neighborhood of non-identical nodes resulting

Table 3.2: Epinions (self-validation): *False Positive* vs. *True Positive* for both schemes

Scheme 1

False Positive	0.01%	0.1%	1%	10%	25%
1-hop	70.81	90.39	96.80	99.38	99.63
1,2-hop	30.35	46.82	67.25	87.15	93.35
2-hop	5.11	17.36	35.32	68.42	84.65
Complete	4.41	17.76	35.67	68.08	83.79

Scheme 2

False Positive	0.01%	0.1%	1%	10%	25%
1-hop	35.08	52.92	82.87	97.33	100.00
1,2-hop	11.37	25.95	62.11	83.95	93.70
2-hop	1.86	7.33	47.71	99.98	100.00
Complete	0.34	5.89	36.33	94.99	98.62

from sampling links out of 1 000 ego-nets rather than 100 as was the case for Epinions. We study the sources of error in § 3.4.5.

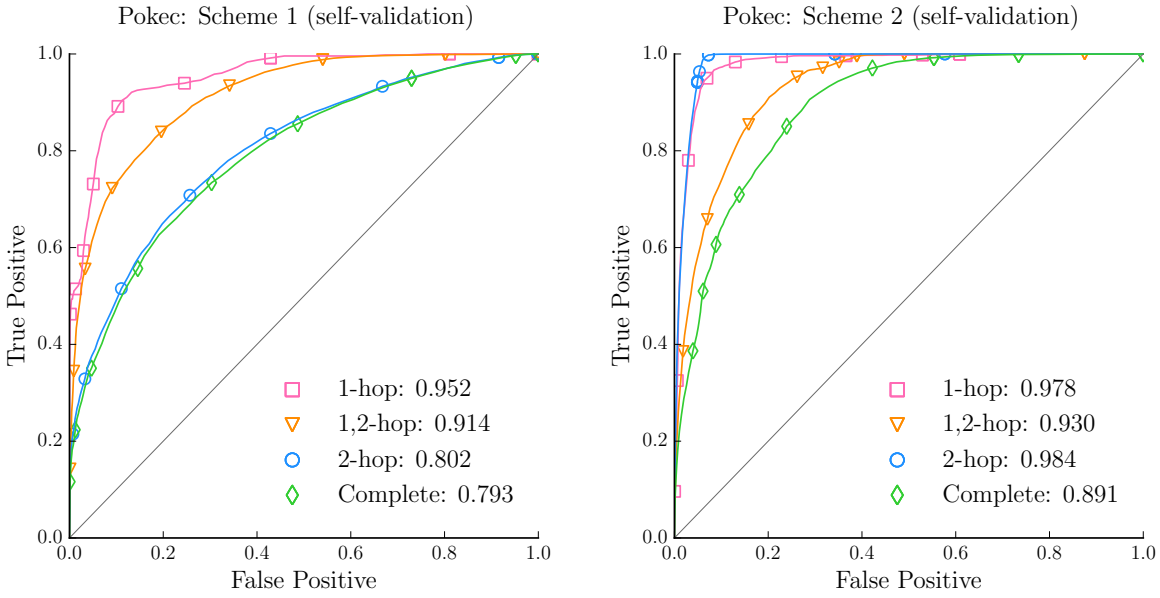


Figure 3.6: Pokec (self-validation): ROC curves for both schemes

3.4.3 Results: different training distribution

In this section we evaluate the performance of the classifiers trained on totally different distributions. For this, we turn the classifier trained on the 100 Epinions ego-nets to

Table 3.3: Pokec (self-validation): *False Positive* vs. *True Positive* for both schemes

Scheme 1

False Positive	0.01%	0.1%	1%	10%	25%
1-hop	27.50	42.92	51.04	88.75	93.96
1,2-hop	5.25	11.58	36.16	73.24	88.68
2-hop	0.00	12.55	23.15	49.14	69.96
Complete	0.01	10.44	20.48	47.60	68.36

Scheme 2

False Positive	0.01%	0.1%	1%	10%	25%
1-hop	4.20	16.26	49.89	97.20	99.58
1,2-hop	0.79	6.41	28.32	73.88	94.66
2-hop	1.62	12.12	50.42	99.96	99.99
Complete	0.68	6.12	21.14	64.12	86.10

classify test data from Pokec (see, Figure 3.7 and Table 3.4), and vice versa, we use the classifier trained on 1 000 Pokec ego-nets to classify the Epinions test data (see, Figure 3.8 and Table 3.5). The two networks are of a totally different nature: Epinions is a small web-of-trust on a consumer reviews site; Pokec is a much larger national social network.

It is clear that a different training distribution has a detrimental effect on the quality of classification for very low false positive values. For a false positive rate of 0.1% Case 1, 2 and 3 links in the Pokec network are linked at a rate of 27.29%, 10.10% and 4.18% respectively for Scheme 1, and 5.40%, 2.08% and 13.57% for Scheme 2 (see, Table 3.4). The results on the Epinions network see a similar fall (see, Table 3.5). Despite this, they never become small enough to guarantee that the likelihood of linking is negligible. For higher false positive rates (1%) the linkage rate becomes significant again, particularly for Case 1 links, with success rates of 34.79% (Scheme 1, Pokec), 12.29% (Scheme 2, Pokec), 53.45% (Scheme 1, Epinions) and 64.51% (Scheme 2, Epinions). The linkage rate for the Pokec network under Scheme 2 is the largest for Case 3 links, namely 13.57% (0.1% FP), and a surprising 45.45% (1% FP).

We conclude that while training on examples from a totally different distribution yields lower true positive rates for equivalent false positive rates, it would not leave either anonymization Scheme 1 or 2 unscathed. If data were to be released, a significant number of links, particularly Case 1 links, could be uncovered with a low error rate. Where the adversary can use side information to build better priors (using additional attributes or known seed nodes as suggested by previous research) a significant fraction of the links would be recovered.

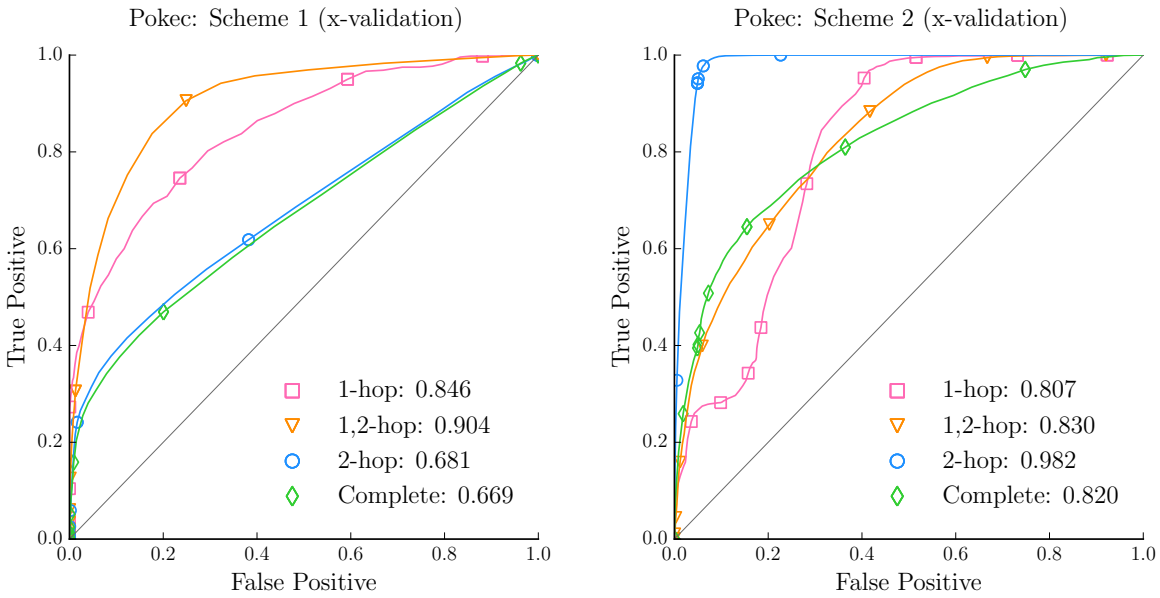


Figure 3.7: Pokec (x-validation): ROC curves for both schemes

Table 3.4: Pokec (x-validation): *False Positive* vs. *True Positive* for both schemes

Scheme 1

False Positive	0.01%	0.1%	1%	10%	25%
1-hop	19.38	27.29	34.79	57.92	76.25
1,2-hop	2.98	10.10	26.52	70.37	90.72
2-hop	1.71	4.18	18.84	39.12	52.52
Complete	1.89	4.05	16.83	36.81	50.76

Scheme 2

False Positive	0.01%	0.1%	1%	10%	25%
1-hop	2.11	5.40	12.29	28.29	60.26
1,2-hop	0.18	2.08	14.34	49.25	70.76
2-hop	3.02	13.57	45.45	99.80	100.00
Complete	1.00	5.61	19.22	56.90	72.76

3.4.4 Traditional de-anonymization task

To assess the generality of the proposed algorithm we apply it to the traditional de-anonymization task, which requires mapping individuals between two different social networks. The adversary uses auxiliary information from a social network to which they have insider access to compromise privacy using the learned mappings from the released network. Narayanan and Shmatikov [5] present an attack based on topology to map node

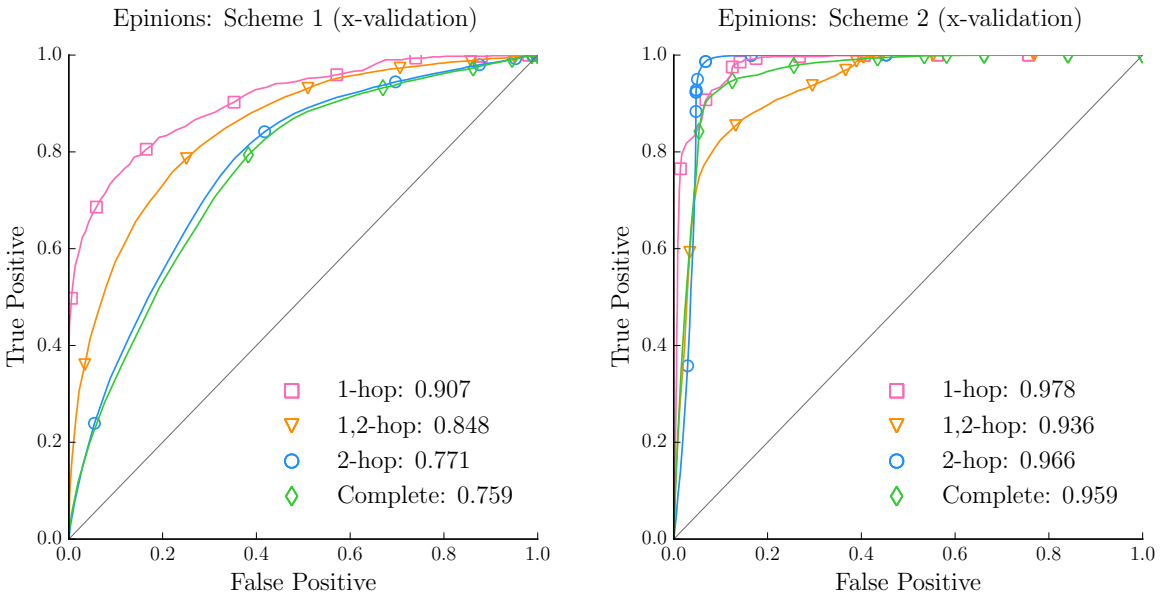


Figure 3.8: Epinions (x-validation): ROC curves for both schemes

Table 3.5: Epinions (x-validation): *False Positive* vs. *True Positive* for both schemes

Scheme 1

False Positive	0.01%	0.1%	1%	10%	25%
1-hop	17.86	43.60	53.45	74.63	85.34
1,2-hop	2.71	6.11	19.79	57.65	78.54
2-hop	0.13	0.68	5.99	35.99	64.20
Complete	0.05	1.87	7.52	33.40	61.54

Scheme 2

False Positive	0.01%	0.1%	1%	10%	25%
1-hop	1.44	6.56	64.51	93.64	99.69
1,2-hop	0.57	3.27	23.80	82.46	91.93
2-hop	0.03	1.17	9.72	99.69	99.99
Complete	0.72	2.96	23.42	93.12	97.75

pairs across social networks that proceeds in two phases: First, the attacker manually maps a few "seed" nodes that are present in both the anonymized target graph and attacker's auxiliary graph; then a propagation phase begins which extends the seed mappings to new nodes based on topology. The new mapping is fed back to the algorithm, eventually resulting in re-identification of nodes across the two graphs.

The performance of the algorithm is evaluated in [5] by synthetically generating auxiliary and sanitized graphs from a real social network. We follow the same technique as described

in detail in § 2.7 to generate an overlapping pair of auxiliary and sanitized graphs.

Given the graphs G_{aux} and G_{san} we trained our classifier to distinguish a node pair as being identical or non-identical. A handful of seed mappings are used to train the learning algorithm. We tweaked the features slightly to take advantage of the global information: we add the 1-hop feature vector (see, § 3.3.3) to the feature vector of the 2-hop neighborhood of the target node to produce a vector twice as long. Decision trees only consider component pairs from the corresponding neighborhood vector to decide the split, i.e. a feature of 1-hop neighborhood is never matched to a feature of 2-hop neighborhood. We use the Flickr social graph [161] (also used in [5]) to generate V_{aux} and V_{san} of size about 50 000 nodes with an $\alpha_V = 0.25$. We train 400 trees with 5 000 non-identical pairs and a varying number of identical pairs (seeds) – 10, 50, 250, 1250, and carry out testing on 10 000 pairs of each class.

Figure 3.9 and Table 3.6 demonstrate that even with as few as 10 seeds we get a significant true positive rate (16.74%) for an appropriately low false positive rate (1%). Increasing the number of seeds and edge overlap improves the results. The re-identification achieved by our classifier is not dependent on a critical mass of seed mappings, large scale re-identification in the algorithm presented by Narayanan and Shmatikov depends sharply on the number of seed mappings. Our classifier degrades gracefully as the number of seeds are decreased. The results are similar even when the classifier is trained with graphs generated using Epinions and then used to attack graphs generated using Flickr, which conclusively indicates that our algorithm can learn de-anonymization independent of the data being analyzed.

We note that the previous results [5] are reported as absolute success or error percentages on a global matching task, rather than a pair-wise matching task, and a specific dataset. They report 30.8% of mappings being re-identified correctly, 12.1% incorrectly, and 57% not being identified. Sadly, we cannot compare the results directly, since we do not perform a global match. Chapter 5 presents a global matching attack derived from our classifier and also compares our attack to Narayanan and Shmatikov [5] as well as several other attacks in § 5.5.

3.4.5 Error analysis

This section investigates a key source of false positives, namely graph overlap between distinct nodes. The features used to characterize and match nodes are based on the degrees of its neighbors. Therefore we expect nodes that share a significant number of friends would be harder to distinguish than those sharing just a few.

To study this hypothesis, we classify 10 000 identical pairs and 10 000 non-identical pairs, that have been anonymized using the edge perturbation technique (with $\alpha_E = 0.25$, 50 seeds, see § 3.4.4). The classifier is tuned to achieve an overall 1% false positive rate,

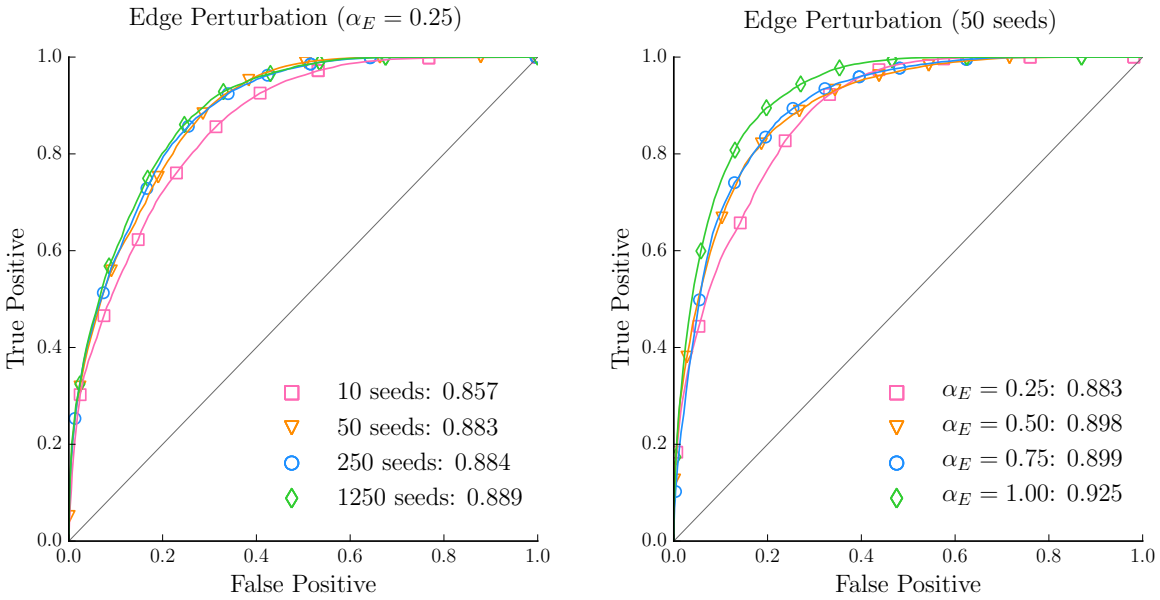


Figure 3.9: Flickr: ROC curves for edge perturbation

Table 3.6: Flickr (edge perturbation): *False Positive* vs. *True Positive*

Varying number of seeds ($\alpha_E = 0.25$)

False Positive	0.01%	0.1%	1%	10%	25%
10 seeds	0.14	3.23	16.74	52.39	78.39
50 seeds	0.72	5.61	22.01	58.38	84.20
250 seeds	3.68	7.19	22.32	58.61	85.14
1250 seeds	0.86	5.97	23.52	60.26	86.43

Varying edge overlap (50 seeds)

False Positive	0.01%	0.1%	1%	10%	25%
$\alpha_E = 0.25$	0.72	5.61	22.01	58.38	84.20
$\alpha_E = 0.50$	0.71	8.66	24.39	66.07	87.76
$\alpha_E = 0.75$	0.16	4.69	17.60	67.85	89.12
$\alpha_E = 1.00$	2.69	8.53	24.64	74.24	93.07

leading to an overall true positive rate of about 20%. Pairs are also categorized according to the Jaccard Coefficient (JC, see Equation (2.1)) of their 2-hop social networks, on which their feature vectors are computed, which provides a degree of social overlap.

Figure 3.10 illustrates that the True Positive and False Positive rates vary widely depending on the social overlap of the tested pairs. When the overlap is small (JC 0.00 – 0.05) the false positive rate is extremely low at 0.31%, but the true positive rate also suffers greatly.

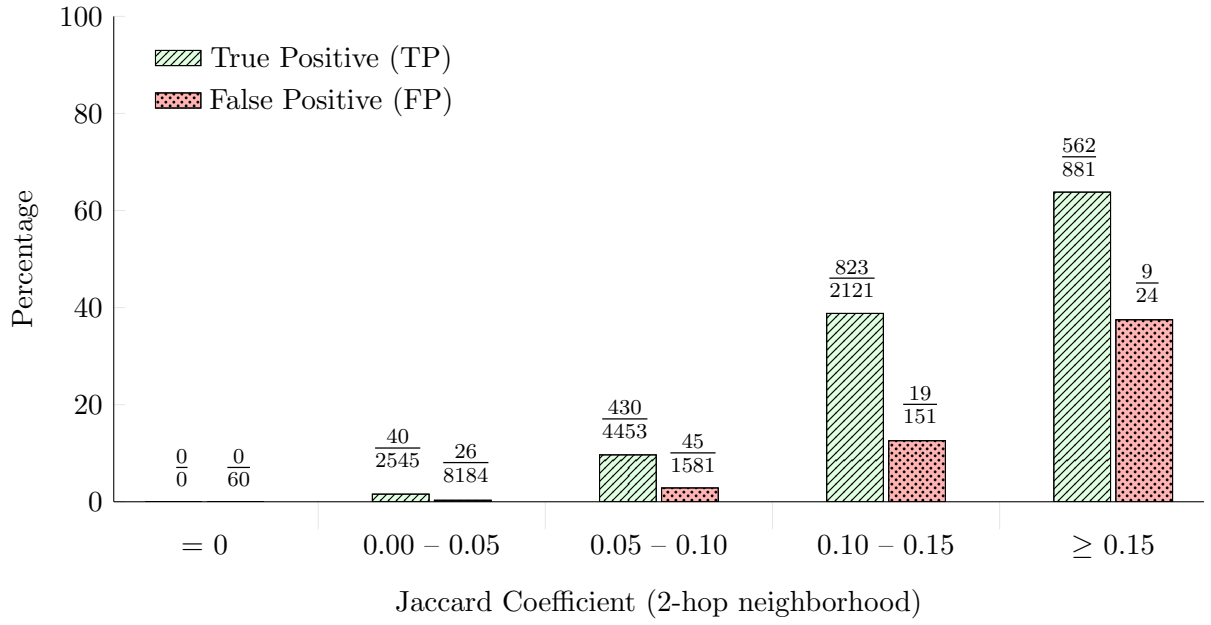


Figure 3.10: True Positive and False Positive rates vary widely depending on the social neighborhood overlap

However, when the overlap is significant (JC ≥ 0.15) both True Positive (63.79%) and False Positive rates (37.5%) rise significantly above the overall baseline. However, only a very small fraction of non-identical pairs have such a high JC (0.24%) , compared to a large fraction of identical pairs (8.81%), leading to a small overall error for this category.

We conclude that False Positives links are likely to occur with nodes in the social vicinity of the actual match. Thus, even erroneous positives allow an adversary to identify the social neighborhood of a match, even if the exact matched node is incorrect.

3.4.6 Data sample sizes

We report the sample sizes for various datasets used in this section. Tables 3.7 to 3.9 show the sample sizes used for evaluation detailed in §§ 3.4.2 to 3.4.4. Table 3.7 depicts the sample size for experiment reported in § 3.4.2 Table 3.1. To run cross classification corresponding sample from the complementary dataset is used. For example to run cross classification for Epinions Scheme 1 for 1-hop node pairs we used 451 Scheme 1 1-hop node pairs sampled from Pokec. When there were fewer than 600 node pairs we trained each tree with all the node pairs available.

3.4.7 Performance

All experiments in this dissertation use code written in Python and run using CPython and a commodity laptop with a 2.8 GHz processor and 16 GB RAM. Time taken to

Table 3.7: Scheme 1: Sample sizes for ad-hoc de-anonymization of Case 1 node pairs

	Sample sizes	
	Identical	Non-identical
Epinions	900	4907
Pokec	1000	5000

Table 3.8: Training and testing number of pairs for Epinions and Pokec

Epinions	**Scheme 1**		**Scheme 2**	
	Train	Test	Train	Test
1-hop	579	812	959	975
1,2-hop	51 716	10 000	11 407	10 000
2-hop	1 910 868	10 000	46 830	10 000
Complete	1 963 163	10 000	59 196	10 000
Non-identical	48 910	10 000	35 655	10 000

Pokec	**Scheme 1**		**Scheme 2**	
	Train	Test	Train	Test
1-hop	451	480	3226	3075
1,2-hop	39 529	10 000	8418	7210
2-hop	1 036 966	10 000	9263	7349
Complete	1 145 419	10 000	20 907	10 000
Non-identical	124 171	10 000	495 353	10 000

Table 3.9: Training and testing number of pairs for Flickr

Varying number of seeds ($\alpha_E = 0.25$)		
	Train	Test
10 seeds	10	10 000
50 seeds	50	10 000
250 seeds	250	10 000
1250 seeds	1250	10 000
Non-identical	5000	10 000

Varying edge overlap (50 seeds)		
	Train	Test
$\alpha_E = 0.25$	50	10 000
$\alpha_E = 0.50$	50	10 000
$\alpha_E = 0.75$	50	10 000
$\alpha_E = 1.00$	50	10 000
Non-identical	5000	10 000

extract features as described in § 3.3.3 is less than six minutes for Scheme 1 and less than a minute for Scheme 2 for both datasets. Table 3.10 shows the time taken to train forests

with 400 trees of each class of identical node pairs and run the classifier for both schemes and datasets. The time taken to test cross classification is similar to the corresponding reported times. Since trees in a forest are independent of each other, training and testing can be run in parallel. However we present the time it takes to perform these operations on a single core. Table 3.11 includes the same details for Flickr edge perturbation strategy.

Table 3.10: Training and testing times for Epinions and Pokec

Epinions	**Scheme 1**		**Scheme 2**	
Time (hrs)	Train	Test	Train	Test
1-hop	14.93	0.86	17.94	1.15
1,2-hop	15.96	1.60	15.93	2.09
2-hop	15.52	1.87	15.44	1.59
Complete	15.58	1.96	16.17	2.08

Pokec	**Scheme 1**		**Scheme 2**	
Time (hrs)	Train	Test	Train	Test
1-hop	10.94	1.00	20.04	1.37
1,2-hop	14.58	2.18	18.15	2.05
2-hop	17.07	2.35	14.56	1.43
Complete	16.80	2.50	18.40	2.36

Table 3.11: Training and testing times for Flickr

Varying number of seeds ($\alpha_E = 0.25$)

Time (hrs)	Train	Test
10 seeds	0.72	0.27
50 seeds	3.21	0.79
250 seeds	5.42	0.97
1250 seeds	14.94	1.28

Varying edge overlap (50 seeds)

Time (hrs)	Train	Test
$\alpha_E = 0.25$	3.21	0.79
$\alpha_E = 0.50$	2.89	0.64
$\alpha_E = 0.75$	3.07	0.66
$\alpha_E = 1.00$	2.88	0.68

3.5 Discussion

We have shown that the machine learning approach to de-anonymization and linkage can be successful. Furthermore, we establish that training can be performed on different graphs than the ones being attacked. Our core evaluation is performed by training on a

small separate set of egonets derived from the same distribution for training and testing. The resulting classifiers are able to de-anonymize egonets with no previously seen nodes. Additionally, classifiers trained on data from a totally different distribution (Epinions and Pokec cross-classification) still perform well enough to classify a non-negligible fraction of pairs as identical, even for low false positive rates. At the very least they can be used to identify a number of common seed nodes, to support further linkage that can tolerate higher false positive rates. We also show that the classifier generalizes well and successfully handles de-anonymization task in the traditional setting. This provides further evidence of its applicability to evaluate novel anonymization schemes applied to diverse datasets. As shown via thorough experiments in Chapter 4, this type of analysis is relevant to the evaluation of any new anonymization algorithm, since as demonstrated, finding suitable graphs and generating training data is easy. Chapter 5 shows the versatility of our classifier by using it to determine a global match across graphs.

Resilience of classifier. The resilience of the classifiers is due to the nature of the training and the classification task: the training algorithm is provided with pairs of nodes, resulting from an anonymization algorithm, and provided labels about whether they are the same or not. As a result it learns invariants that are characteristic of the anonymization method, not merely the data. One might naïvely conjecture that basing the node feature vectors on neighborhood degree distribution would be inept in attacking anonymization schemes that perturb the node degree at random (see, § 3.4.4). The success of classification is not conditioned on the "invariance" of the node degrees but on their "variance" as a function of the anonymization strategy. This function can be learned and hence hedged to attack the scheme. We conjecture that purely synthetic specially crafted graphs may be equally (or even better) suited to train the classifiers, but we leave this investigation for future work.

About half a dozen of the previous attacks in the literature gave a significant inspiration to the work described here. Backstrom *et al.* [87] presented active (sybil) and passive attacks based on searching for patterns in subgraphs to learn relationships between pre-selected target individuals of an anonymized social graph. Group membership has been shown to be sufficient to identify an individual in a social network [117].

Narayanan and Shmatikov [5] present a de-anonymization attack on a social network using auxiliary information from a different social network. They were also the first to note that a large volume of matching errors are in the vicinity of the actual matching nodes. Similar techniques were used to attack the Netflix dataset by correlating it with IMDb dataset [3]. We test our algorithm in this setting but also highlight some key differences that makes their approach hard to apply to the D4D dataset. The algorithm proposed aims to link two large correlated social networks. However, the D4D dataset is split into small ego-nets. Consequently, one cannot straight-forwardly apply a method based on identifying few known seed users and expanding from them, since such a propagation would naturally end

at the boundaries of each ego-net. For this reason our learning approach does not make use of known seeds, but instead we use training examples that do not necessarily comprise the nodes in the subgraph to be de-anonymized. This allows the proposed approach to train on different graphs, and still link nodes in small subgraphs of other networks. However, we note that when a large subgraph is available the two approaches may be combined: one may use the techniques proposed in this chapter to identify few seeds with high certainty, and then apply our techniques to identify known nodes in their vicinity, which are more likely to be related than random nodes.

Narayanan *et al.* [4] combine graph de-anonymization and random forests to obtain very good results for link prediction. Cukierski *et al.* [158] showed that comparable results for could be achieved by using pure random forests for link prediction, rather than de-anonymization, for the same dataset. In contrast, we use the techniques for identity reconciliation, we do not have directionality available for our graphs and our feature extraction is simple and efficient, an important factor for attacking huge datasets.

Our work uses these techniques for the first time in an adversarial de-anonymization setting: whereas previous de-anonymization techniques considered graphs that were organically noisy but structurally intact. In contrast the D4D challenge organizers purposefully and aggressively alter graph topology to prevent linkage.

3.5.1 Is anonymization effective?

So are anonymization Schemes 1 & 2 effective? First, it is clear that Scheme 2 is mildly more effective than Scheme 1. Our attack only considers nodes with degree over five, hence, for a given egonet Scheme 1 exposes more nodes to our attack than Scheme 2 due to lower damage to the node degrees. Also, the true positive rate of the classifier is lower for any fixed acceptable false negative rate. However, even for extremely low false positive rates of 0.01% a non-negligible fraction of the nodes are correctly classified ranging from 1.86% for 2-hop nodes to 35.08% for 1-hop nodes (Epinions) and from 1.62% to 4.20% (Pokec). First, a person may with non-negligible probability be within a successfully linked pair. Second, extremely high confidence matches can be used to piece together disparate egonets into larger graphs. Those larger graphs, with some common nodes, can be then further used to de-anonymize other users.

Therefore we believe that the linking rates we observed are too high for the original social network to be effectively unrecoverable. We conclude that the D4D competition organizers were prudent to limit the disclosure of the dataset to known participants and require contractual assurances that they would not de-anonymize the data.

3.5.2 Improving de-anonymization

The D4D data release provides access to egonets observed between different time slots, with same identifiers across time slots (see, § 3.2.1). We can amplify our classification success by classifying a candidate node pair in each time slot and then applying a majority rule for deciding the true association – identical or non-identical. Such an attack is significantly more potent than one possible on an aggregated social graph across all time periods. The original D4D dataset also contained edge weights. We chose to ignore those to devise de-anonymization strategies for generic graphs without such weights. However, the feature vector proposed could be augmented to take those weights into account to generate distinct features.

A key difference between the proposed de-anonymization algorithm, and previous work [5], is the lack of reliance on known "seeds". Those seeds are adversary side-information, i.e. a few known nodes between two unlinkable networks, that can be used to unravel the anonymization. Our algorithms take two totally distinct egonets, for which no common node is known, and classify nodes within them as identical or non-identical. In case some seeds are known, our techniques can be applied to the common neighbors of the seeds to determine whether they are the same node or not. In that case larger false positive rates can be tolerated since the a-priori probability of the nodes being the same is larger (by many orders of magnitude) compared to any two random nodes. Since the proposed approach works extremely well for larger false positive rates (like 1%) such a combined scheme is extremely effective as shown in Chapter 5. A full investigation of the iterative application of the presented method, to build up a full graph is beyond the scope of this work.

3.5.3 Choosing tree parameters

In this section we discuss the choice of parameters to train decision trees. The parameters directly affect predictive accuracy, accuracy of the confidence, its generalization and computational complexity of training and testing. Below we discuss how we chose the factors that have the strongest influence on the results.

Forest size. Literature suggests [147, 162, 163] that testing accuracy increases monotonically with the forest size T. Using a small forest size decreases the accuracy of uncertainty and produces low quality confidence estimates, this leads to erroneous generalization. Criminisi *et al.* [148] present examples in their report which obtain good results for $T = 400$, hence we have chosen the forest size to be 400. We tested our results with $T = 500$ without any appreciable gain.

Randomness. We inject randomness while training using bagging and random node optimization. Generating trees randomly produces different trees that lead to smooth

decision boundary for the decision forest. Randomization leads to a lower confidence in the posterior, hence we obtain smoother and more spread out posteriors. The optimal separating surface for each weak learner is less sharply defined for the forest and the effect of entire training data overpowers the effect of individual training points. We use bagging by passing 600 node pairs (per class) sampled at random from the training data set (Table 3.8, § 3.4.6) to each tree. Every split node is exposed to 5% of the total feature parameters. We experimented by increasing the percentage to 10%, 20% and 25% but did not observe any benefits.

Features. We use the feature vector length as 70 and bin size as 15. The intuition behind this choice is that most individuals in a social network have fewer than 1 000 friends and the above choice reflects this. We study the impact of vector length and bin sizes by training a random forest with $T = 200$ and bagging with 200 node pairs (per class) for Epinions Scheme 2 (complete set of node pairs). We experimented with vector lengths 21, 35, 105 and bin sizes 50, 30, 10 respectively. Increasing the vector length increases the complexity by a quadratic factor. Figure 3.11 and Table 3.12 demonstrate that the performance is similar for different vector lengths. The vector length of 21 performs the best for the demonstrated example but we found that our choice of vector length 70 works best over all across multiple settings.

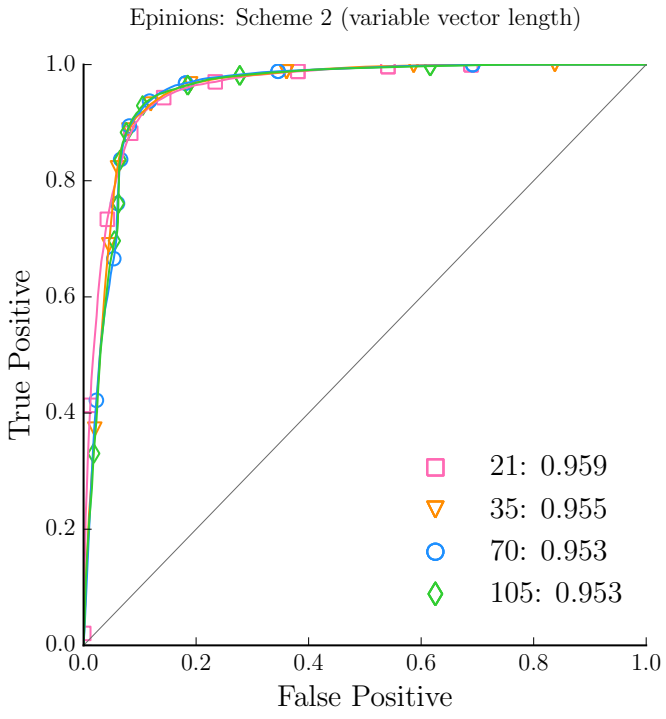


Figure 3.11: Epinions (Scheme 2, complete): ROC curves for effect of vector length

Our classifier can be further improved by weighting the *important* features using two forests. The features selected by the first forest are chosen with a higher probability than the rest of the features to train a second forest. The literature indicates that this produces slightly better results, we do not use this technique in our work.

Finally, we experimented with D4D by selecting features beyond the 1-hop neighborhood of the node pair under consideration. This sharply increased our false positive rates. This is due to the fact that many nodes share large parts of their 2-hop neighborhood, and the number of features in that space is much larger than than in the 1-hop neighborhood. This is in contrast with the de-anonymization features used in § 3.4.4. This happens because the global properties of the overlapping subgraph are better preserved when the full graph is available instead of just the local neighborhood. This allows us to select features from the diverse 2-hop network without a sharp increase in false positives. Picking parameters to be optimal across all settings is hard; instead we focus on developing techniques that perform well for appropriately chosen parameters.

It may be the case that a biased proposal of features to train split nodes may overcome this problem, and even produce better results. The inability of the weak learner to infer the position of the pair of nodes via-a-vis the ego (as ego, 1-hop or 2-hop) also illustrates that non-local properties of the subgraphs may contain useful information to improve classification accuracy.

Table 3.12: Epinions: Varying vector length (Scheme 2 - Complete): *False Positive* vs. *True Positive*

Varying feature vector length

False Positive	0.01%	0.1%	1%	10%	25%
21 X 50	1.07	11.16	37.62	90.88	97.35
35 X 30	0.40	3.78	22.97	91.45	97.75
70 X 15	0.49	1.49	21.01	91.87	97.99
105 X 10	0.49	2.39	21.38	92.37	97.82

Training. Posteriors obtained from the forest can be controlled by varying the ratio of data points of the training classes. Increasing the ratio for a particular class increases the confidence of prediction and accuracy for the class by imposing a higher penalty for incorrect classification of the over represented category. We tested this by changing the ratio of input node pairs to 1:10 in the favor of non-identical node pairs. This decreased the false positive rate hugely with an expected increase in the prediction confidence at the cost of lowering the true positive rate. This option is available to attackers dealing with very large datasets, and therefore requiring a truly minute false positive rate – however, all our training is performed with an equal number of identical and non-identical node pairs.

3.6 Summary

In this chapter we demonstrated how graph de-anonymization can be automated using structural features to identify nodes. The technique presented is robust to noise and generalizes across a variety of graphs and anonymization techniques. The model uncovers artefacts and invariants of any black-box anonymization scheme from a small set of examples. Despite a high degree of automation, classification succeeds with significant true positive rates even when small false positive rates are sought. We show next in Chapter 4 how these techniques can be leveraged to evaluate novel anonymization techniques quickly and cheaply.

Chapter 4

Benchmarking social graph anonymization schemes

4.1 Introduction

Social graphs provide a rich source of data for analyzing and studying a variety of human behavior. As a result the demand for real world datasets in academia and industry has always been high. There is also the possibility of social good coming from such datasets due to insights that it might provide. Such benefits, however, come at a price. As per prior discussion in § 2.1.1 releasing rich datasets documenting private behavior of individuals has the potential to cause a privacy catastrophe. To alleviate these concerns, data providers often choose to scrub off personal identifiers and claim this is enough to preserve privacy. Such superficial scrubbing has been known to be less than ideal [3–5], even though it may provide the advantages of simplicity and legal compliance (or perhaps plausible deniability in case of a privacy breach).

Anonymizing high-dimensional data is a very hard problem and conventionally it is considered unwise to publish graph data even without identifiers. A sizable amount of effort has been spent in devising better social graph anonymization schemes that preserve privacy without hindering analysis [82, 89, 93–95], however without providing much assurance. It should be noted that some information needs to be destroyed to protect privacy and this always affects analysis: there is an inherent tension between preserving privacy vs. preserving utility and one adversely affects the other. But what exactly is the trade-off? Most schemes are proposed in an ad-hoc manner and show no incremental evolution, thus confounding their comparison. This has resulted in a large number of anonymization schemes, but as shown in Chapter 3, constructing attacks even for simple schemes can require careful study and manual work. This has created a skewed ecosystem where anonymization algorithms can be proposed without much effort and a considerable time must be spent to evaluate them.

Hence it has become imperative to devise a yardstick to measure and compare anonymization schemes against one another – a benchmark. Having a standard benchmark would allow us to rank the schemes based on the anonymity provided and utility preservation, starting form an equal footing. To this end we propose a machine-learning framework to benchmark perturbation-based social graph anonymization schemes. To the best of our knowledge this is the first attempt to compare and benchmark anonymization schemes in a quick and automated manner. The framework can efficiently handle any perturbation-based social graph anonymization scheme as well as a wide variety of adversarial models.

Although the node features presented can easily accommodate node and edge attributes we focus on structure-based node re-identification. Graph structure is important for studying social behaviors and interconnectivity of the entities make social graphs unique as a dataset. Consequently, most data providers publish graph structure in some form and the bulk of the de-anonymization research has been concentrated in proposing schemes to obfuscate it. Hence, the analysis presented in this chapter focuses on graph structure-based attack to evaluate anonymization schemes. Chapter 3 cast social graph de-anonymization as a learning problem; this chapter extends and builds upon the model developed to benchmark perturbation-based social graph anonymization schemes. In this chapter we demonstrate the diversity of learning task and model developed by analyzing a variety of anonymization schemes.

Our contributions. In this chapter we make the following primary contributions:

- We design and implement a machine-learning framework to benchmark perturbation-based social graph anonymization schemes. The framework provides a quick and automated platform to evaluate and compare anonymization schemes efficiently (§ 4.2.3).
- We present a mechanism to train the framework without ground truth by generating subgraphs and sampling training data from the auxiliary and sanitized social graphs (§ 4.2.3).
- We conduct a thorough analysis of the effect of graph perturbation on the anonymity achieved and utility preserved using publicly available real world social graphs. To this end we analyze six popular graph perturbation schemes including those promising k-anonymity based on two real world social networks for three perturbation levels each – a total of 36 configurations (§ 4.3).
- We conduct a thorough analysis of the effect of graph perturbation on utility by analyzing five fundamental graph metrics for each of the 36 configurations (§ 4.3). We examine whether the anonymization schemes provide any anonymity at all even when perturbation levels are high enough to destroy almost all utility.

The work presented in this chapter is based on the paper titled *True Friends Let You Down: Benchmarking Social Graph Anonymization Schemes* [164] published in the Proceedings of the 9th ACM Workshop on Artificial Intelligence and Security (AISec 2016).

4.2 Quantifying anonymity in social graphs

Quantifying anonymity in social graphs is hard. Even for a given anonymity scheme it is challenging to quantify the relation between anonymity and graph perturbation. It is much harder to compare the anonymity provided by different anonymization schemes. All meaningful schemes are constrained by preserving utility thus making them vulnerable to attack. We exploit the utility constraint to construct a learning algorithm that learns features from anonymized graphs to devise de-anonymization attacks.

4.2.1 The adversarial model

As shown in Chapter 3, even after node identifiers have been removed the structure of a social graph can be used to splice it with overlapping social graphs thus revealing scrubbed-off identities and potentially private relations among them. Ideally an anonymization scheme should render relinking attacks discussed in Chapter 3 impractical while still retaining the utility of datasets.

As discussed in Chapter 2, the graph anonymization literature contains a wide variety of schemes with varying aims. Comparing these schemes requires a common adversarial model. To measure the efficacy of an anonymization scheme we quantify the success rate in re-identifying common nodes in an overlapping pair of graphs, both of which have been anonymized using the scheme being analyzed. The adversary uses one of the graphs as background knowledge to attack the other graph. This is similar to the setting used by Narayanan and Shmatikov [5] to measure the success of their re-identification algorithm. One of the other popular adversarial models [82–84, 165] assumes that adversary knows the target degree, this is unrealistic but easy to evaluate hence popular. We cannot consider a weaker adversarial model because attacks [5] already exist under the current adversarial model. We focus on structure based attacks as most of the graph perturbation based schemes are designed to conceal the graph structure and prevent re-identification based on node neighborhood. However, due to its modularity the adversarial model and learning model can incorporate a variety of adversaries which are much more advanced. We describe our adversarial model in greater detail in the subsequent sections.

4.2.2 Graph generation

Benchmarking an anonymity scheme begins with generating a pair of graphs with an intersecting set of nodes from real world graphs [5]. Each graph is anonymized using the scheme to be benchmarked. We treat one graph as the target graph (mimicking the sanitized version released) and the other graph as the auxiliary graph at attacker's disposal (used as side information for linking identities), these can be interchanged. We constrain the attacker to only have the knowledge of graph structure. This is the least amount of information which is released and inclusion of any other information such as node or edge attributes only strengthens the attacker.

We produce two overlapping graphs G_1 and G_2 as detailed in § 2.7; in our experiments we use $\alpha_V = 0.25$. G_1 and G_2 are subsequently anonymized using the scheme to be benchmarked (instead of `RSP` as fixed earlier), the anonymized auxiliary and sanitized graphs thus produced are called G_{aux} and G_{san} respectively. Creating overlapping graphs allows us to quantify the efficacy of the anonymization scheme in preserving privacy by measuring the success of structure-based re-identification. The efficacy is measured in terms of success of the classification model in differentiating whether two individuals belonging to social graphs G_{aux} and G_{san} are identical or not.

4.2.3 The classification framework

We propose a machine-learning classification framework that uses structure-based re-identification to measure the privacy leak in social networks. The classifier can quantify anonymity of social graph anonymization schemes. As presented in Chapter 3, the framework is based on an ensemble of randomly trained decision trees known as random decision forest [148]. The forest is trained to classify node pairs (n_{aux}, n_{san}) such that $n_{aux} \in V_{aux}$ and $n_{san} \in V_{san}$ as identical or non-identical using the features of each node.

Features

As detailed in § 3.3.3 we use the degree distribution of the network around a node to derive features to represent it. Full graphs have much diverse neighborhoods than egonets hence we utilize both the 1-hop and 2-hop node neighbors to construct a node's feature vector. Both 1-hop and 2-hop features are computed separately and then concatenated. Figure 4.1 shows a sample feature vector quantizing the number of nodes with degree in a given range. In our experiments we chose the individual vector length to be 21 and bin size to be 50.

We also use the Silhouette Coefficient of degrees of a node pair as a feature which is used in parallel with degree distribution to decide the split in a tree node. The Silhouette

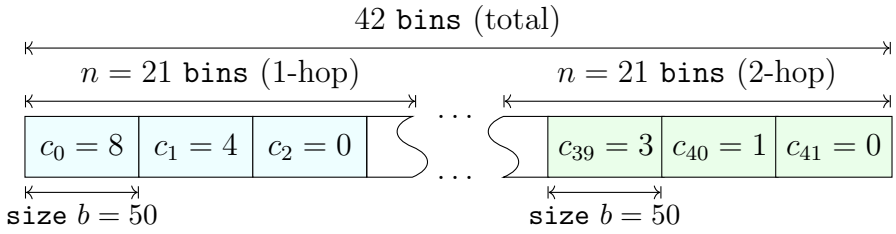


Figure 4.1: Example node feature vector

Coefficient of degrees of two nodes belonging to G_{aux} and G_{san} is defined as:

$$\delta(d_1, d_2) = \frac{\mid d_1 - d_2 \mid}{\max(d_1, d_2)}$$

where $d_1 = \text{degree}(n_{aux})$, $n_{aux} \in V_{aux}$ and $d_2 = \text{degree}(n_{san})$, $n_{san} \in V_{san}$.

We note that the modularity of these features allows us to tune them according to the adversary's background knowledge. For instance if the graph was directed and directionality was part of the adversarial model then the features could be expanded to contain in-degree and out-degree of 1-hop and 2-hop neighborhoods instead of combining the degrees.

Training

We present an improvement over the seed dependent training shown in § 3.4.4 (which was used earlier solely to conduct a shoulder to shoulder comparison with Narayanan and Shmatikov [5]) by using a mechanism to train a model in the absence of ground truth or seed mappings across auxiliary and sanitized data. In the scenario of a graph release, an adversary does not possess the real data but a morphed and damaged version of it. The adversary attempts to splice it with the data at his disposal to re-identify individuals. Training a machine learning model is tricky in such a scenario as we need access to the ground truth. To re-identify nodes with high confidence it is important to train the model with a substantial amount of high quality data. Ideally, it would be best to train a model by generating auxiliary and sanitized graphs from the same graph that was used to generate G_{aux} and G_{san}. Providing the adversary access to such data makes it too strong and unrealistic. We get around this problem by training the classifier using node pairs from G_{aux} and G_{san}, both the graphs are split again to produce two sets of overlapping graphs. This time however, we do not apply any anonymization on the generated pairs as they have already undergone anonymization. This allows us to sample node pairs from the overlapping sets for which we know the ground truth. Data is sampled from each set and merged to train the classifier as demonstrated in § 3.3.4. The adversary only needs a rough estimate of the overlap between auxiliary and sanitized graphs to sample data in the manner described thus producing node pairs which closely resemble those being attacked.

We also experimented by sampling training data by generating auxiliary and sanitized graph from a different but similar social network to that under attack. The learning task is transferable [159] thus cross training works as well but we get better results by training from a distribution which is as close to the original as possible. Splitting G_{aux} and G_{san} to train simulates training under ideal circumstances as closely as possible. G_{aux} and G_{san} represent a damaged version of the original graph (G_1 and G_2 respectively), however, under our adversarial model they are the closest dataset to that being attacked and hence we only need to split them as they have already undergone perturbation. Training the forest allows it to learn features that optimize the classification success. A random forest is comprised of trees that are trained by randomly sampling training data and node features thus helping it to learn the features for classifying previously unseen node pairs. We use a forest of 400 trees in our experiments.

Classification

After training the decision forest a pair of feature vectors (v_{aux}, v_{san}) representing the node pair (n_{aux}, n_{san}) sampled from G_{aux} and G_{san} is passed through the forest as outlined in § 3.3.4. Each tree in the forest assigns a probability to the pair of being identical or non-identical. After the node pair has passed through all the trees we average the predictions to reach a final prediction.

4.3 Evaluation and results: anonymity vs. utility

We benchmark graph anonymization schemes based on two properties – quality of anonymization and utility preservation. For each of the six social graph anonymization schemes – `RSP`, `RAD`, `RSW`, `REP`, `KDA` and `1HKA` (see, § 2.6), we measure how de-anonymization success and utility vary versus strength of anonymization. Intuitively, if an increase in anonymization does not produce a commensurate decrease in de-anonymization success while substantially diminishing utility then such an anonymization scheme is considered less favorably. All the schemes being evaluated provide varying levels of *anonymity* by controlling the level of graph perturbation. We note that increasing perturbation does not necessarily provide more anonymity in all cases but it always affects utility adversely.

We evaluate the anonymization schemes using two publicly available real world social graph datasets, Flickr [161] – a popular social network for sharing pictures and Facebook New Orleans dataset [166] – a dataset extracted from the world's most popular online social network. The graphs for evaluation are generated as defined in § 4.2.2. The original Flickr graph has 80 513 nodes and 5 899 882 edges, the graphs generated for benchmarking have about 50 000 nodes and 2 310 000 edges prior to any anonymization. The original Facebook graph has 63 731 nodes and 817 090 edges, the graphs generated for benchmarking have about 40 000 nodes and 320 000 edges prior to any anonymization.

Interpreting the ROC curves. For a given social graph anonymization scheme and anonymity strength, each node pair passed through the classification framework is assigned a score which is a real number in [0, 1]. This procedure is carried out for more than a million randomly selected node pairs to analyze the success of structure-based re-identification. The score assigned to each node pair signifies its likelihood of being non-identical. An ideal classifier will output 1 whenever it sees a non-identical node pair and a 0 whenever it sees an identical node pair. The Receiver Operating Characteristic (ROC) curves illustrate how close the classifier is to an ideal one. It does so by measuring the True Positive (TP) rate as the False Positive (FP) rate tolerated is varied in the range [0, 1]. An ideal classifier gives a TP rate of 1 at FP rate 0, whereas TP and FP are always the same for random guessing. In practice a classifier will always make errors (FP), our goal is to maximize the correct classification rate (TP) for the error tolerated. The Area Under the Curve (AUC) provides a summary of the quality of the classifier, an ideal classifier has an AUC = 1 whereas random guessing produces a classifier with an AUC = 0.5.

Measuring Anonymity. Anonymity of a scheme is measured by the de-anonymization success achieved as depicted by ROC curves and the AUC (shown in the figure legend); this allows us to compare schemes. We also compare the performance of the classifier when the graph is simply split (denoted as GS) and no anonymization is applied (node pairs are sampled from G_1 and G_2) to the case where a particular scheme is used (node pairs are sampled from G_{aux} and G_{san}).

Measuring Utility. Measuring utility is harder as there is no standard metric to capture it. An anonymization scheme such as `RSW` might perfectly preserve the degree distribution of a graph while damaging other properties. Also, utility depends on how the analyst uses data. We look at some fundamental utility metrics as they vary with anonymization level. The properties we study are:

- Degree distribution – it measures the frequency of degrees as they grow and it is an important measure of a small world graph.
- Joint degree distribution – the distribution of node degree pairs between which edges exist in a graph.
- Average degree connectivity – the average nearest neighbor degree of nodes with degree d.
- Degree centrality – the fraction of total nodes a node is connected to.
- Eigenvector centrality – it measures a node's importance based on the its connection to other important nodes. For a vertex v it is defined as the v^{th} component of the

eigenvector associated with the largest eigenvalue of the adjacency matrix of the graph.

These properties are fundamental to the behavior of social graphs and damaging them significantly has an adverse effect on the overall utility of the graph. Most of the usage specific graph properties such as diameter, average path length, clustering co-efficient, closeness centrality, betweenness centrality, pagerank, link prediction, number of connected components etc. cannot be conserved without conserving the properties studied. For instance all centrality and pagerank metrics are tightly coupled with eigenvector centrality. Accurate measurement of these metrics is important for studying community structure, designing routing algorithms and ranking problems. High levels of perturbation to node degrees affects clustering co-efficient, average path length and diameter of the graph; whereas joint degree distribution is important for link prediction. Due to the interconnected nature of graphs it is very hard to selectively perturb some metrics while conserving others, on the contrary perturbation that conserves particular metrics is very damaging to others. As observed, `RSW` (see, § 4.3.3) preserves the degree distribution perfectly but is very detrimental to other metrics. Figure 4.2 shows the joint degree distribution of unperturbed graphs for reference.

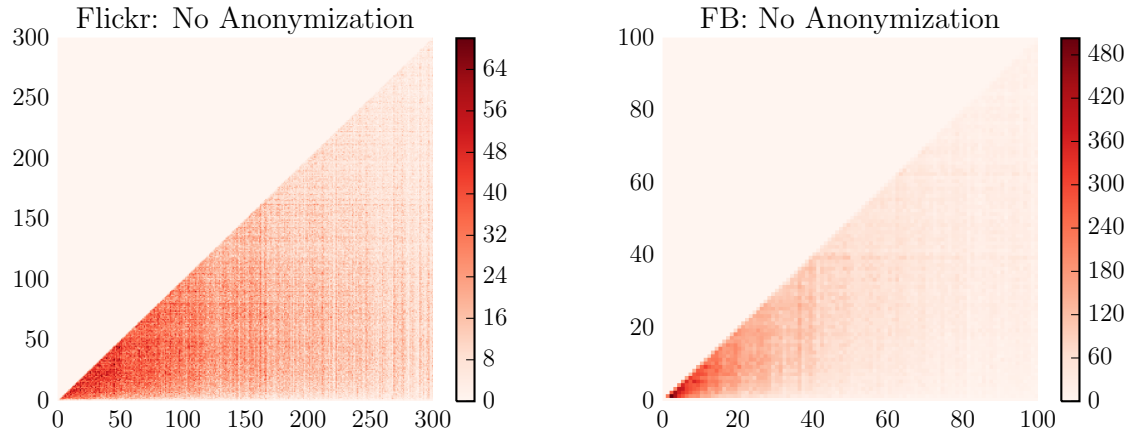


Figure 4.2: Original joint degree distribution

Implementation. The random forest classifier is trained using about 25K identical and 500K non-identical node pairs. We classify 16K identical node pairs for Flickr and 10K for Facebook against 1 Million non-identical node pairs for both graphs. All identical nodes pairs with degree over five are included for classification. We omit the low-degree nodes as they have too little information to be identified reliably; as a result the number of identical node pairs is lower for Facebook which is sparser of the two graphs. Training the classifier takes about 10 minutes while classification takes about 25 minutes.

4.3.1 Random Sparsification (RSP)

Anonymity. Deleting edges at random limits the scope of structural de-anonymization due to lack of information. However, it does not rule out attacks completely. As shown in Figure 4.3 the classification success diminishes with decreasing edge overlap (as measured by Jaccard Coefficient, see, Equation (2.1)). We introduce an edge overlap of $\alpha_E = (0.75, 0.50, 0.25)$ by increasing the fraction of randomly deleted edges to produce G_{aux} and G_{san}; α_E is computed for the common subgraph of G_{aux} and G_{san}, the edge overlap for the entire graph is much lower. Even after lowering α_E to 0.25, enough information remains and de-anonymization is quite successful for both the graphs.

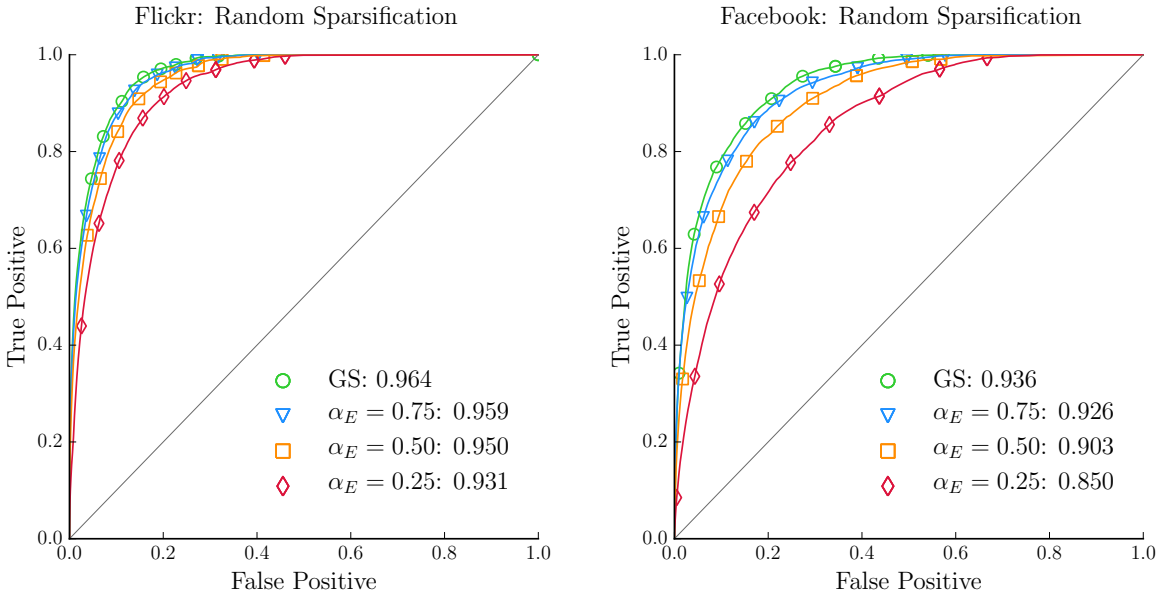


Figure 4.3: RSP: ROC curves

Utility. Deleting edges shows gradual and graceful decline in the quality of the graph.

- Degree distribution (Figure 4.4) – more of less preserved for varying levels of anonymity, this is expected as edges are deleted uniformly at random.
- Joint degree distribution (Figure 4.5) – shifts towards the low degree node pairs as deleting edges decreases the number of high degree nodes.
- Average degree connectivity (Figure 4.6) – shifts towards the origin due to decrease in node degrees across the graph. Decrease in the range of degrees shrinks the spectrum.
- Eigenvector centrality (Figure 4.7) – deleting edges uniformly does not effect the eigenvector centrality much, the important nodes continue to be important. However, the decrease of degree does shift the spectrum.

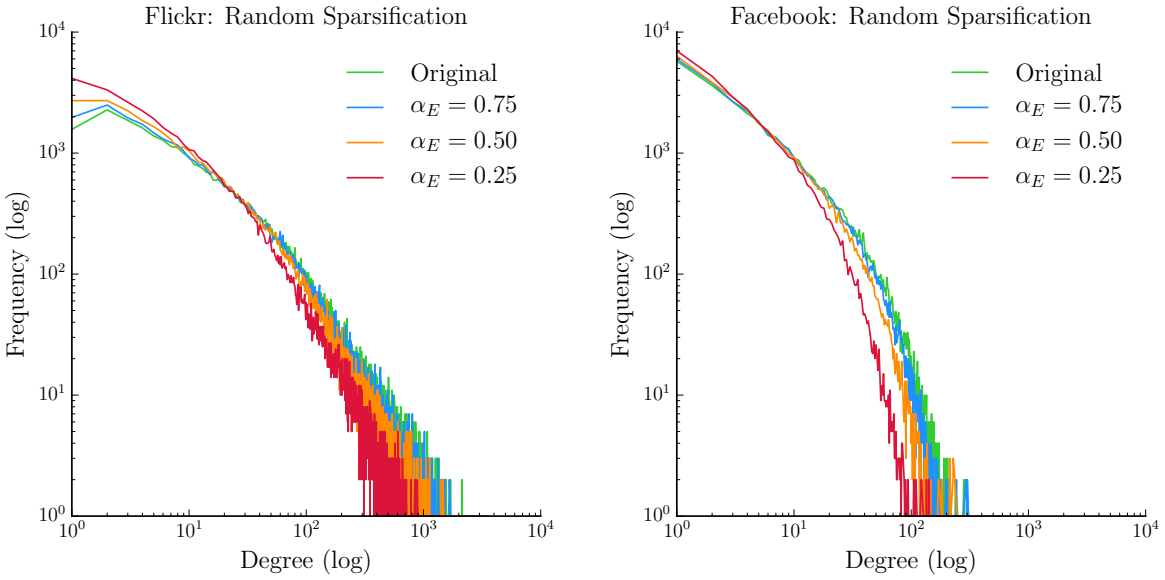


Figure 4.4: RSP: Degree Distribution

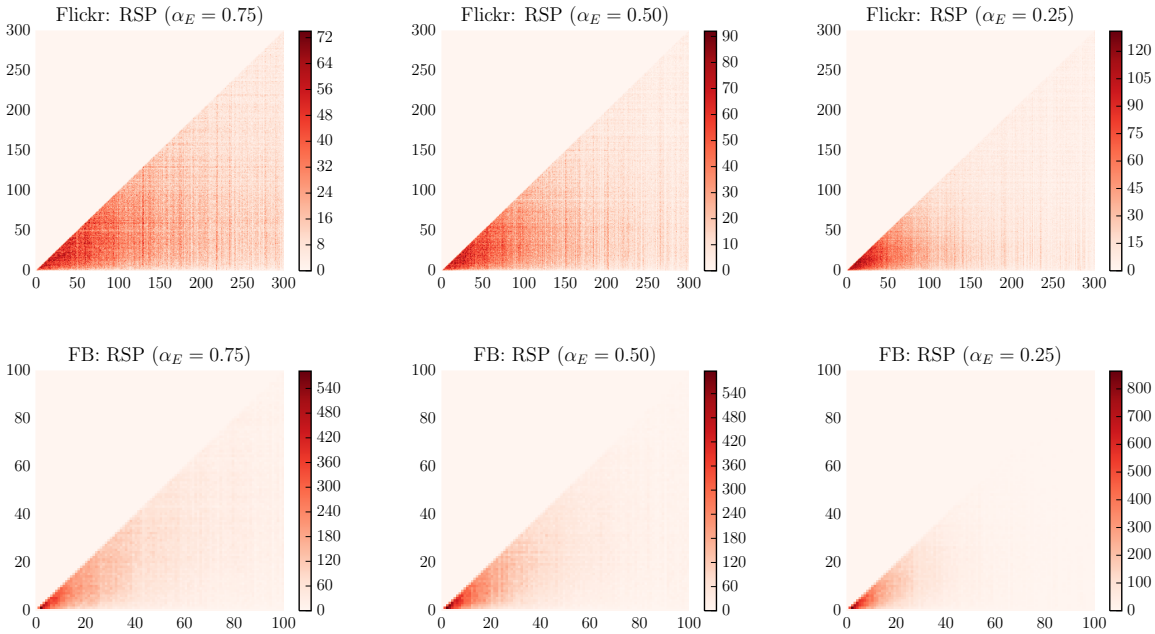


Figure 4.5: RSP: Joint Degree Distribution

4.3.2 Random Add/Delete (RAD)

Anonymity. Adding and deleting edges at random homogenizes the graph by decreasing the degree of high degree nodes and increasing the degree of low degree nodes. Introducing non-edges increases the degree of low degree nodes as non-edges are more likely exist between such nodes. As a result structural classification becomes difficult, as can be seen in Figure 4.8. We compare the performance of our classifier by introducing graph perturbation of a fraction $k = (0.10, 0.25, 0.50)$ of the edges. Even at $k = 0.50$ structure-based de-anonymization is quite successful for both the graphs.

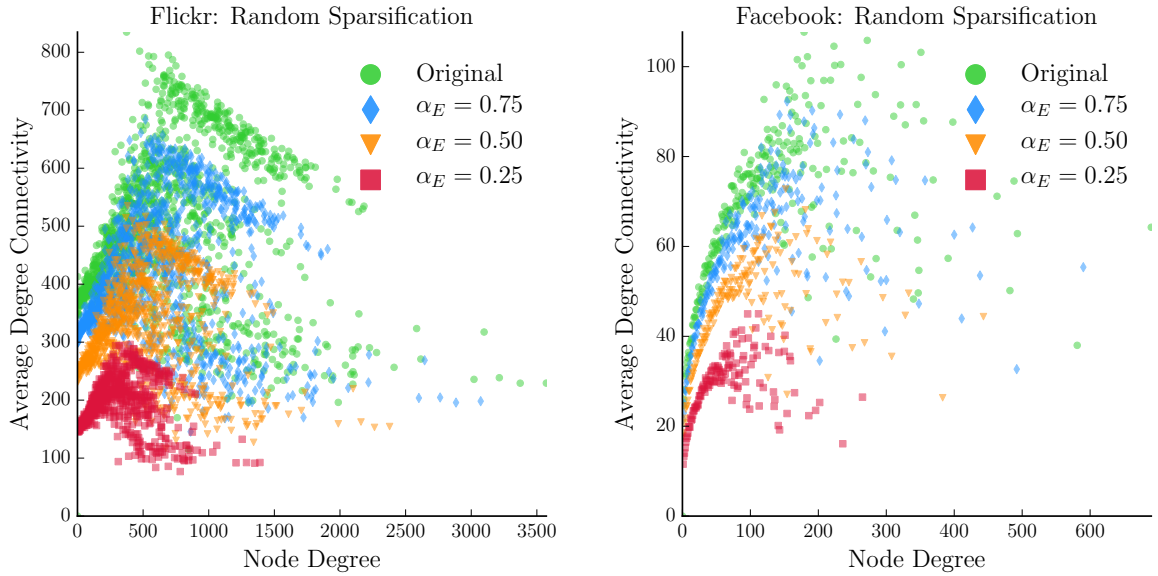


Figure 4.6: RSP: Degree Connectivity vs. Node Degree

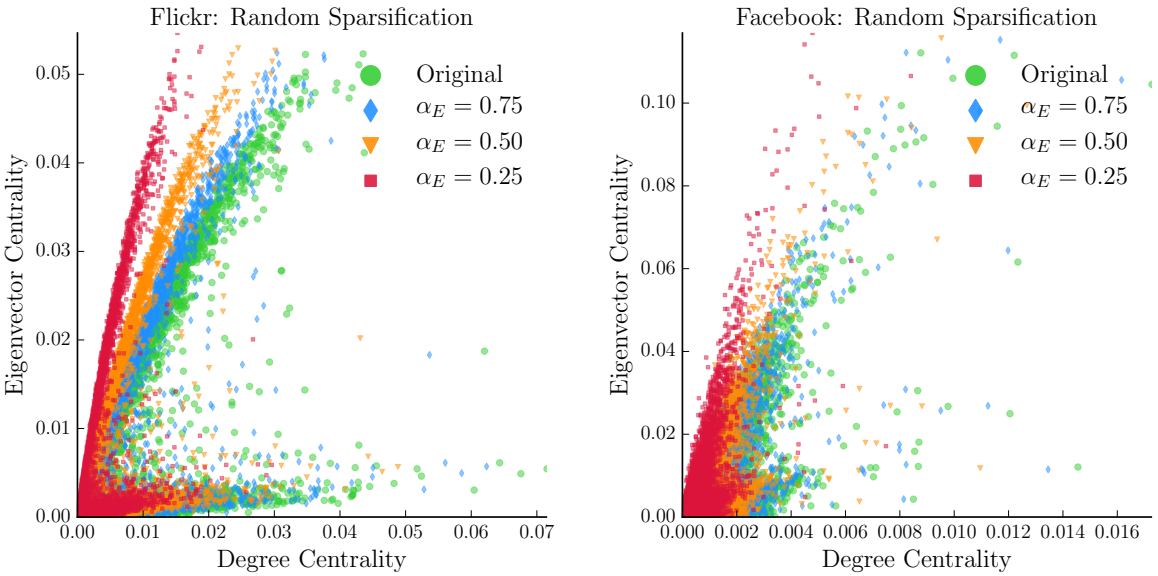


Figure 4.7: RSP: Eigenvector Centrality vs. Degree Centrality

Utility. `RAD` is less forgiving of the graph properties, specially its degree and joint degree distribution. Adding and deleting edges at random pushes the graph closer to a random graph (achieved at $k = 1$).

- Degree distribution (Figure 4.9) – addition and deletion of edges at random makes the degree distribution more compact. The shift is disproportionate due to nature of perturbation being biased.
- Joint degree distribution (Figure 4.10) – confirms that all the node degrees move close together.

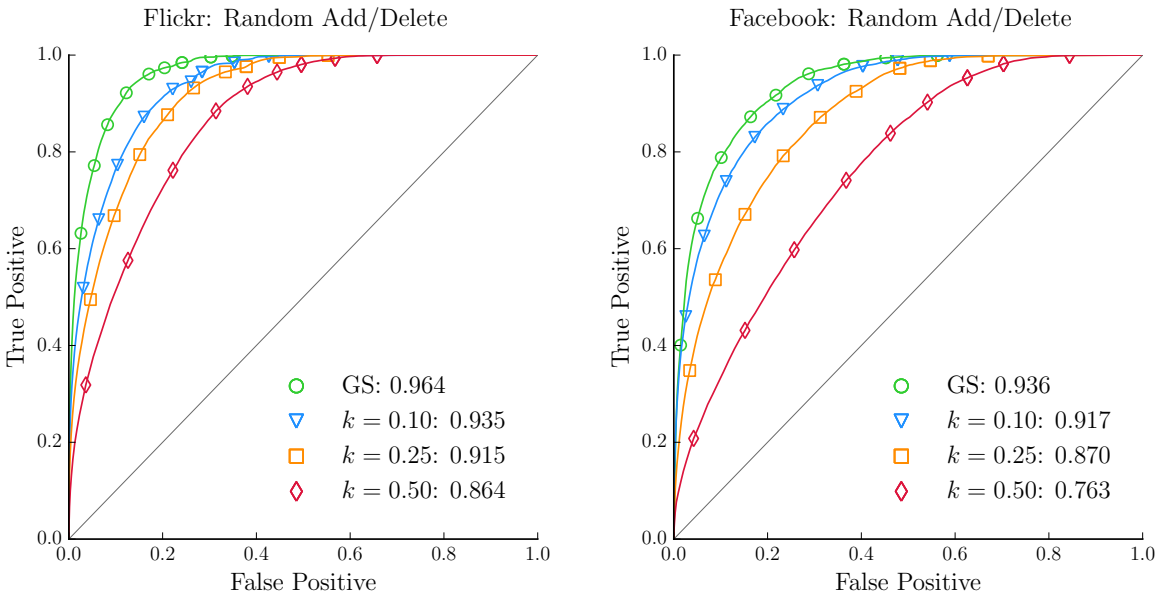


Figure 4.8: RAD: ROC curves

- Average degree connectivity (Figure 4.11) – shifts downwards as low degree nodes get delinked from high degree nodes and linked to other low degree nodes, the high degree nodes suffer a decrease in their degree and loss of links to other high degree nodes, this decreases their connectivity as well. Since node degrees become uniform the spectrum shrinks.

- Eigenvector centrality (Figure 4.12) – perturbation of the neighborhood of high degree nodes decreases their degree; however, they still retain their importance as deleting and adding edges at random does not have a huge effect on their influence. On the other hand low degree nodes still remain relatively unimportant because of being primarily connected to other low degree nodes thus producing the shift observed.

4.3.3 Random Switch (RSW)

Anonymity. Switching edges at random affects the graph properties in an unpredictable manner. Although RSW perfectly preserves the degree distribution, it is rather damaging to the other graph metrics. This scheme provides a perfect example of the challenges faced in preserving utility in a perturbed graph. In our experiments we introduce perturbations of a fraction $k = (0.20, 0.50, 0.85)$ of the number of edge pairs; smaller values of k produced no perceptible change in ROC curves hence larger values are picked to study variance. Figure 4.13 shows that even at the highest level of perturbation barely any additional privacy is achieved. Preserving the degree distribution not only adversely affects the graph's properties but also makes it more vulnerable to structure-based re-identification.

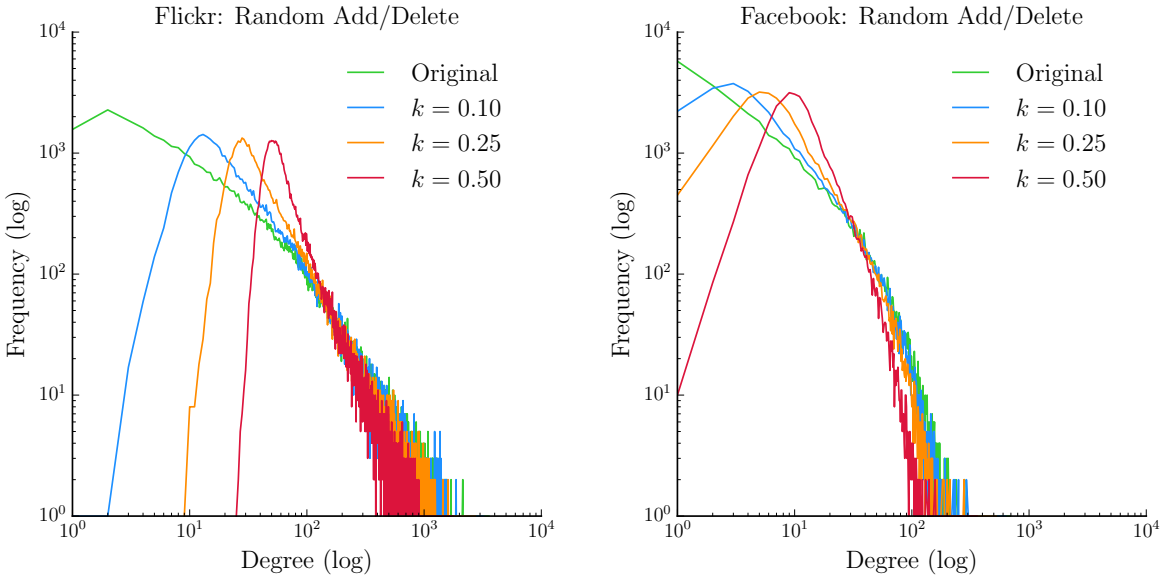


Figure 4.9: RAD: Degree Distribution

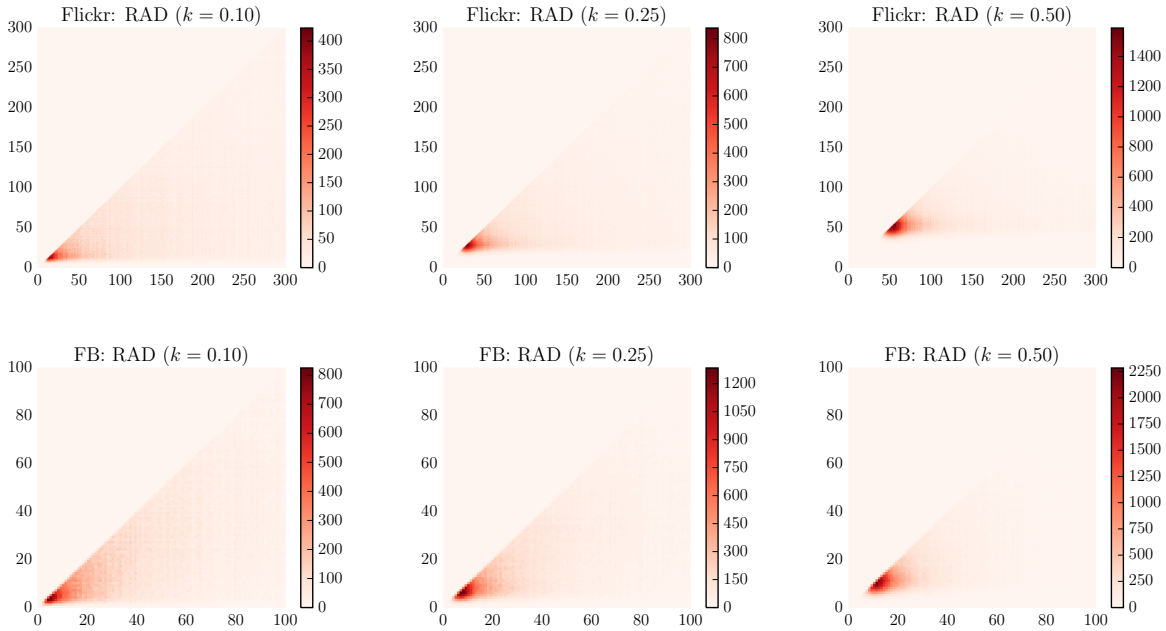


Figure 4.10: RAD: Joint Degree Distribution

Utility. Almost all graph metrics are profoundly damaged apart from degree distribution which is perfectly preserved, although at a high cost.

- Degree distribution – exactly preserved.
- Joint degree distribution (Figure 4.14) – becomes more uniform throughout the graph. The original joint degree distribution (Figure 4.2) shows that edges are concentrated among low degree nodes. Switching edges gradually spreads the concentration towards higher degree nodes.

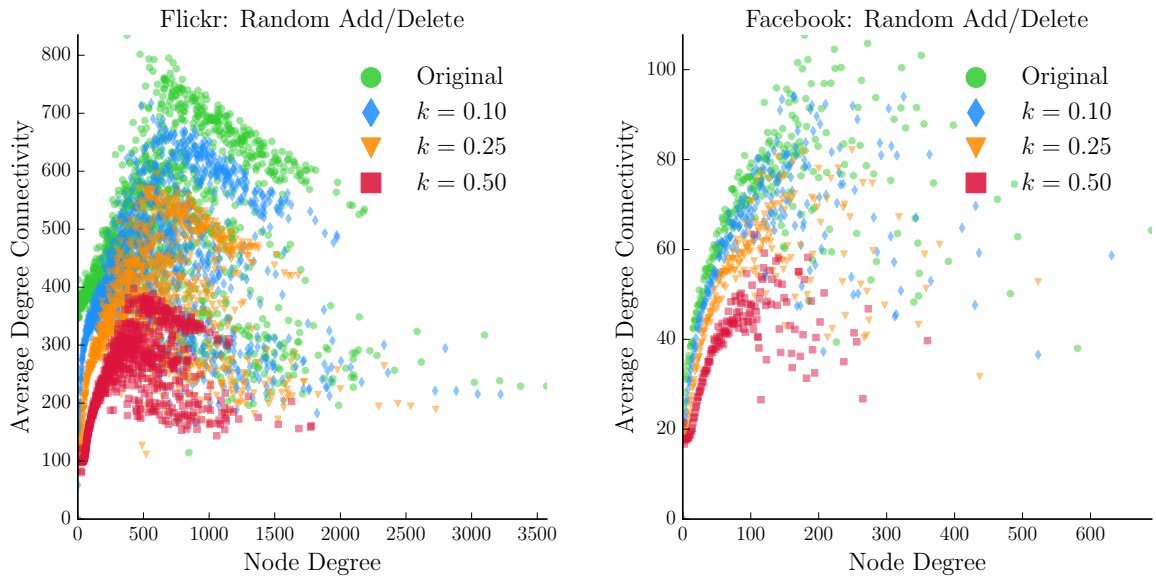


Figure 4.11: RAD: Degree Connectivity vs. Node Degree

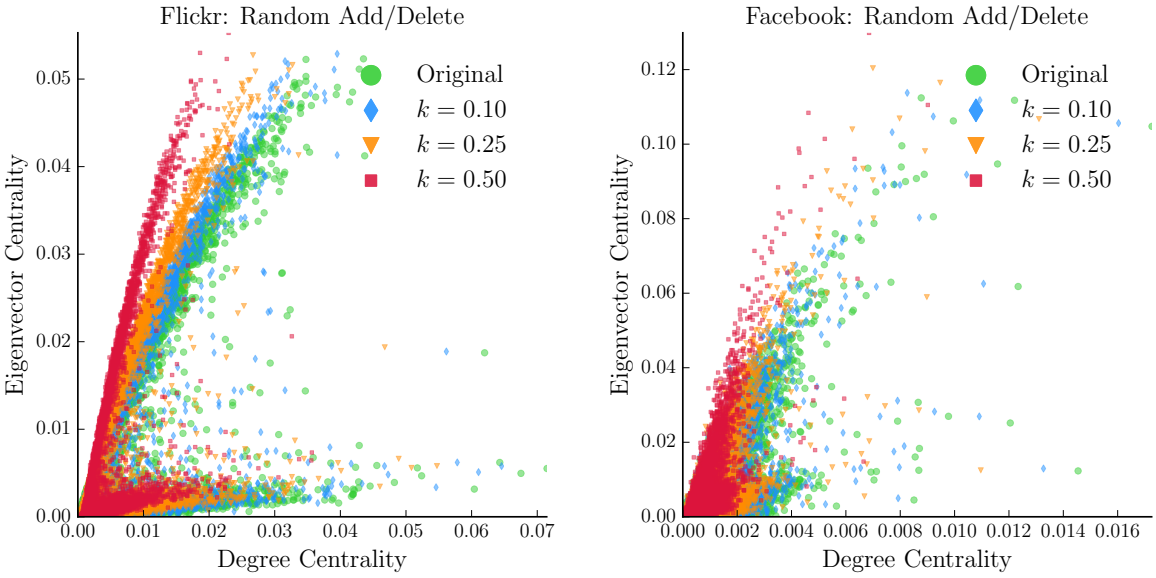


Figure 4.12: RAD: Eigenvector Centrality vs. Degree Centrality

- Average degree connectivity (Figure 4.15) – becomes uniform as low degree and high degree nodes get linked.

- Eigenvector centrality (Figure 4.16) – remains preserved for low levels of perturbation ($k = 0.10$) but the influence of nodes becomes directly proportional to degree centrality at the highest level of perturbation. This is an indication of the graph losing structure and moving towards randomness.

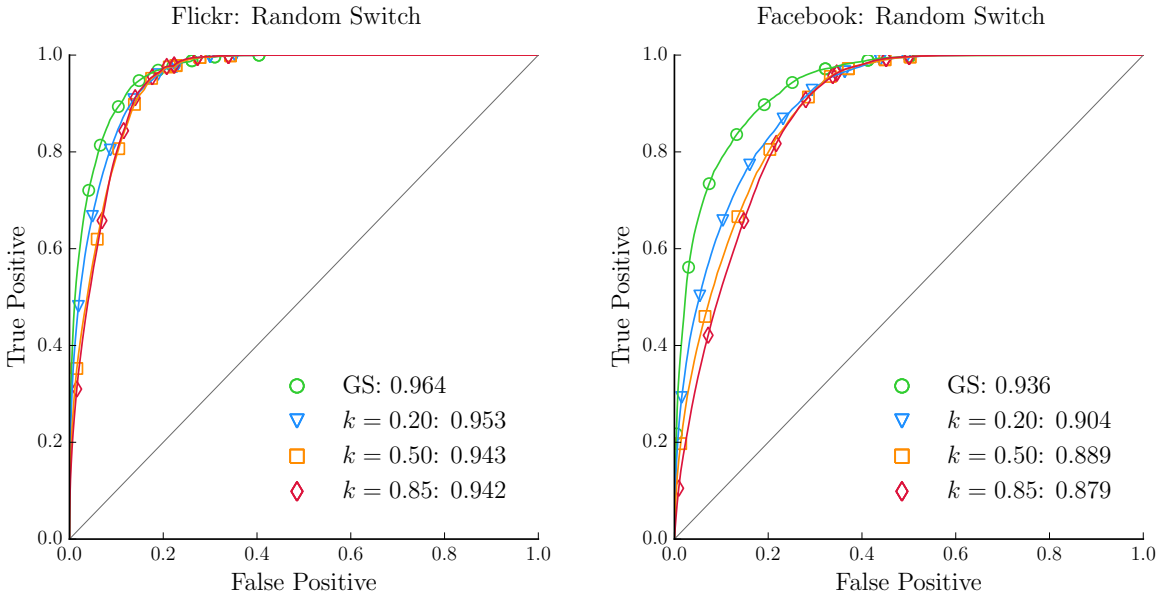


Figure 4.13: RSW: ROC curves

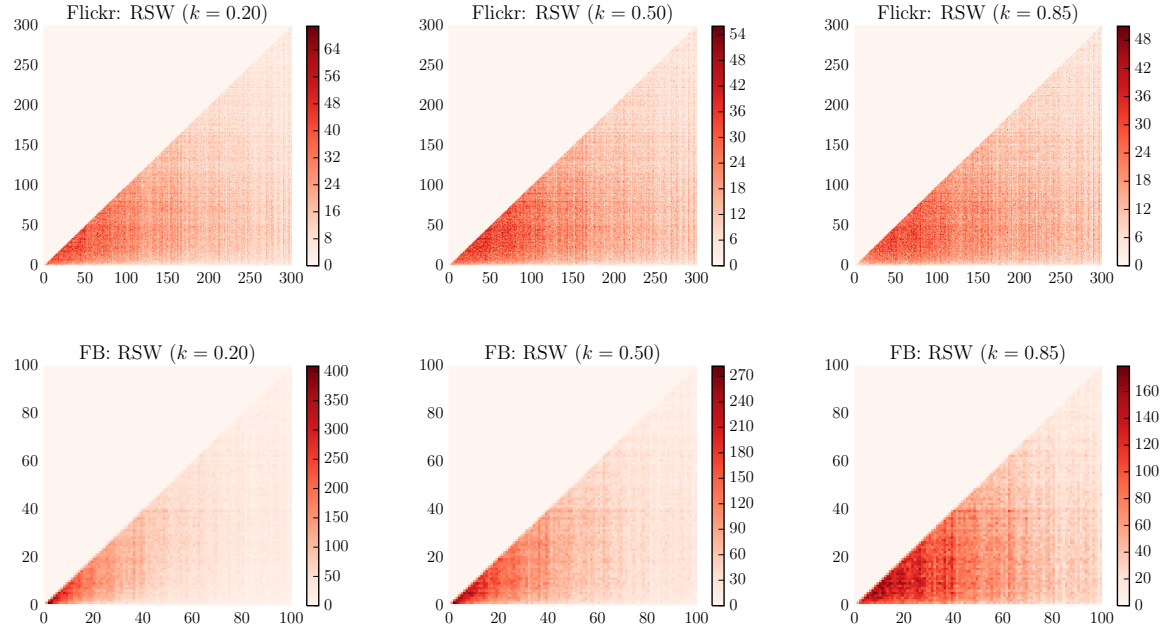


Figure 4.14: RSW: Joint Degree Distribution

4.3.4 Random Edge Perturbation (REP)

Anonymity. Deleting a fraction of edges and adding the same fraction of non-edges produces a large increase in edges overall since the number of non-edges is several orders of magnitude higher than edges. We introduce perturbations of $\mu = (10^{-4}, 10^{-3}, 10^{-2})$ for our analysis. The authors who proposed this scheme [89] consider $\mu = 10^{-3}$ to be a high level of perturbation which indeed it is. However, we did not see any appreciable change in the ROC curve for such low values for Flickr so we raised it further. As seen in Figure 4.17 denser graphs are more resilient to introduction of noise; this is even more

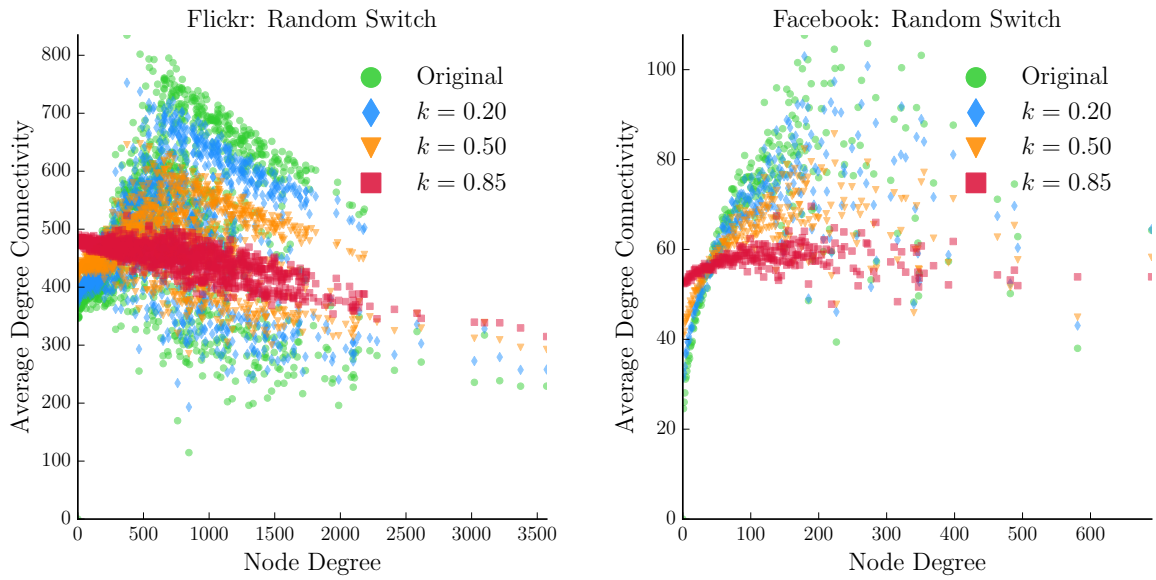


Figure 4.15: RSW: Degree Connectivity vs. Node Degree

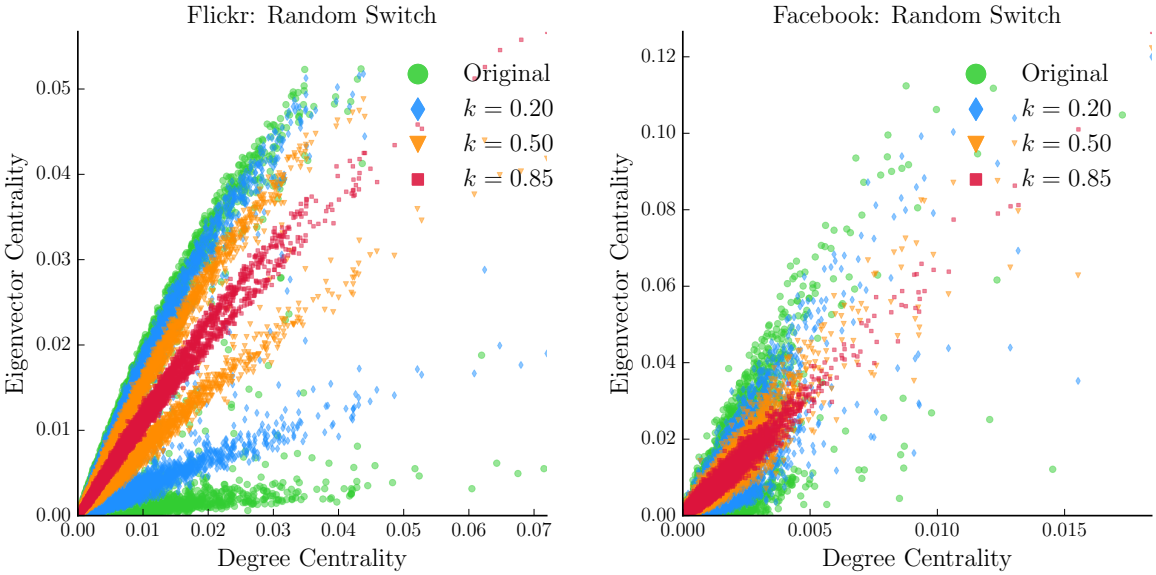


Figure 4.16: RSW: Eigenvector Centrality vs. Degree Centrality

apparent in the case of REP which is rather damaging to Facebook at $\mu = 10^{-3}$ but does little damage to Flickr.

Utility. Although we achieve a reasonable level of anonymity at $\mu = 10^{-3}$ for Facebook it comes at the cost of utility which is significantly damaged. Flickr resists longer but to achieve anonymity we need to compromise utility. At $\mu = 10^{-2}$ both graphs are perturbed beyond recognition and of little use – but even at this level of perturbation Flickr still looks attackable with AUC = 0.792, whereas classifying node pairs in Facebook gets reduced to as good as guessing with AUC = 0.585.

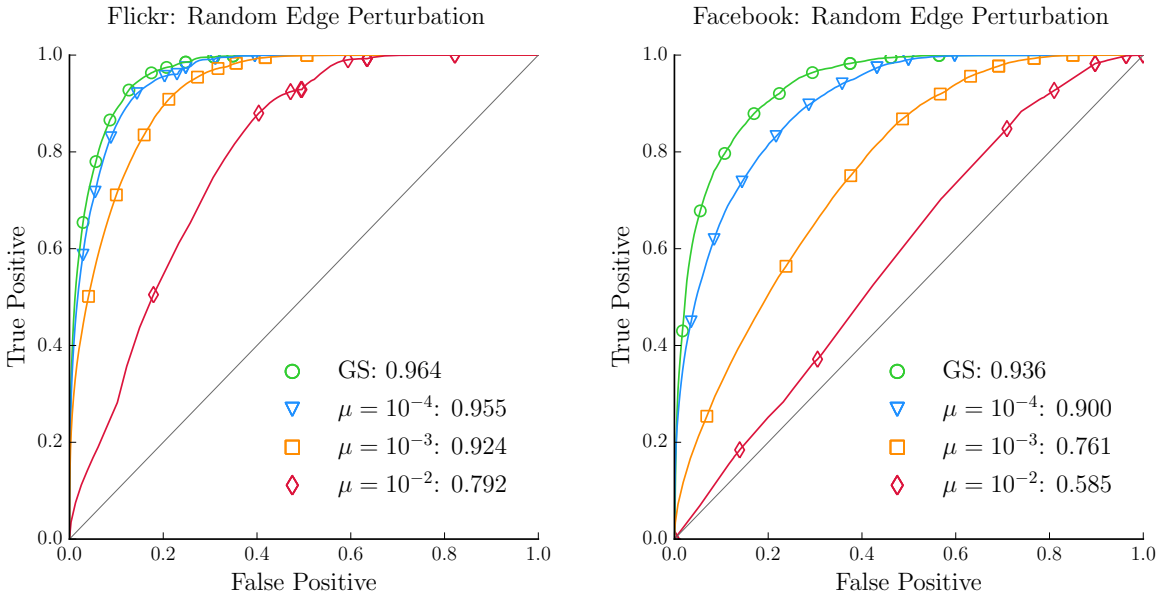


Figure 4.17: REP: ROC curves

- Degree distribution (Figure 4.18) – shifts towards the right but the change is more extreme as the proportion of non-edges introduced is orders of magnitude higher.
- Joint degree distribution (Figure 4.19) – gets concentrated towards the high-degree node pairs.
- Average degree connectivity (Figure 4.20) – addition of edges produces new low degree nodes that are connected to other low degree nodes thus producing a dip of connectivity spectrum towards the origin. The neighborhood of high-degree nodes shows relatively less change.
- Eigenvector centrality (Figure 4.21) – in contrast to `RSP` adding non-edges at random does not change the degree centrality or eigenvector centrality of nodes much. Random edges do not change the importance of nodes, hence we see the spectrum being preserved. However, extreme perturbation ($\mu = 10^{-2}$) causes the spectrum to shift.

4.3.5 k-Degree Anonymization (`KDA`)

Anonymity. Ensuring k-degree anonymity for nodes requires introducing edges among high-degree nodes since such nodes are rarer in the graph. We generate k-degree anonymous graphs using the `Supergraph` and `Greedy_Swap` algorithm [95]. `Supergraph` produces a k-anonymous graph for a given number of nodes after which `Greedy_Swap` swaps edges among node pairs till the generated graph has almost the same edge set as the original graph, in our experiments the generated graph has at least 90% of the same edges as

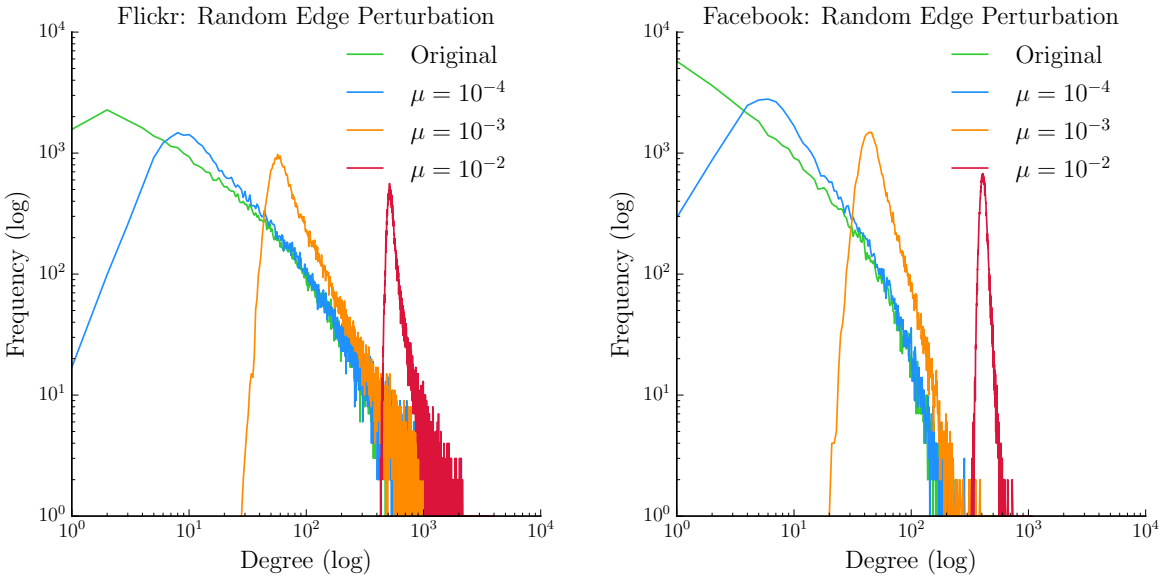


Figure 4.18: REP: Degree Distribution

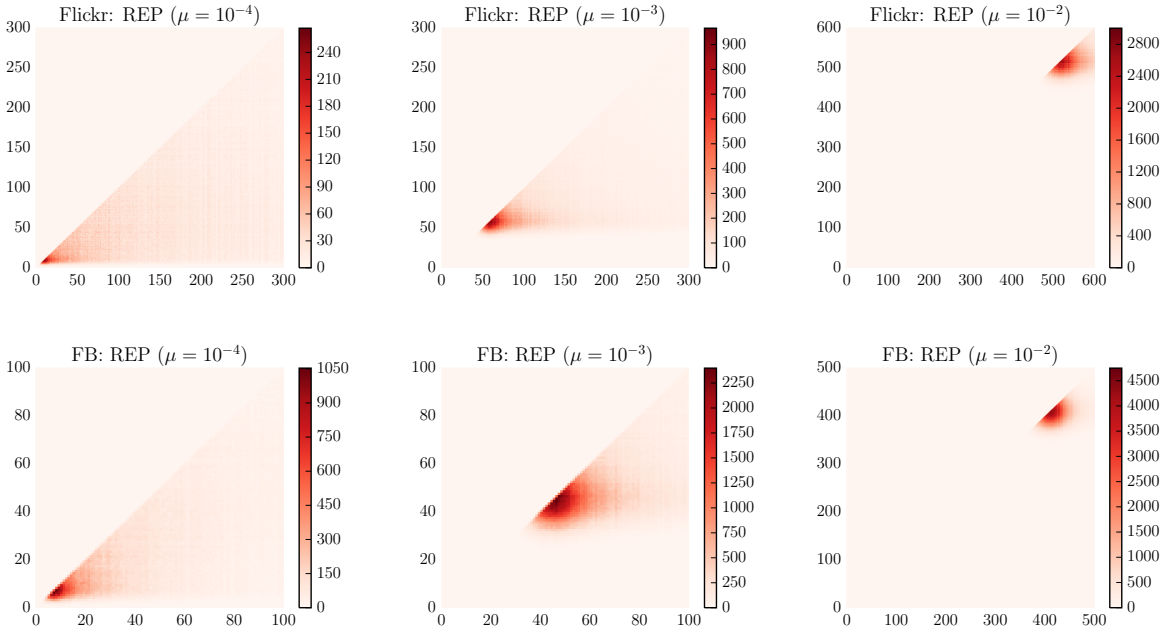


Figure 4.19: REP: Joint Degree Distribution

the original graph. We introduce perturbations of $k = (10, 50, 100)$ for our analysis. As observed (Figure 4.22) even high values of k does not increase the anonymity by much for either graph. Edge insertion among high-degree nodes is not large enough to mask their true neighborhood structure whereas the low degree nodes are left almost untouched. The result is significant damage to graph metrics without gaining much anonymity.

Utility. Change in the neighborhood of high-degree nodes adversely affects the properties of graph without purchasing any additional anonymity.

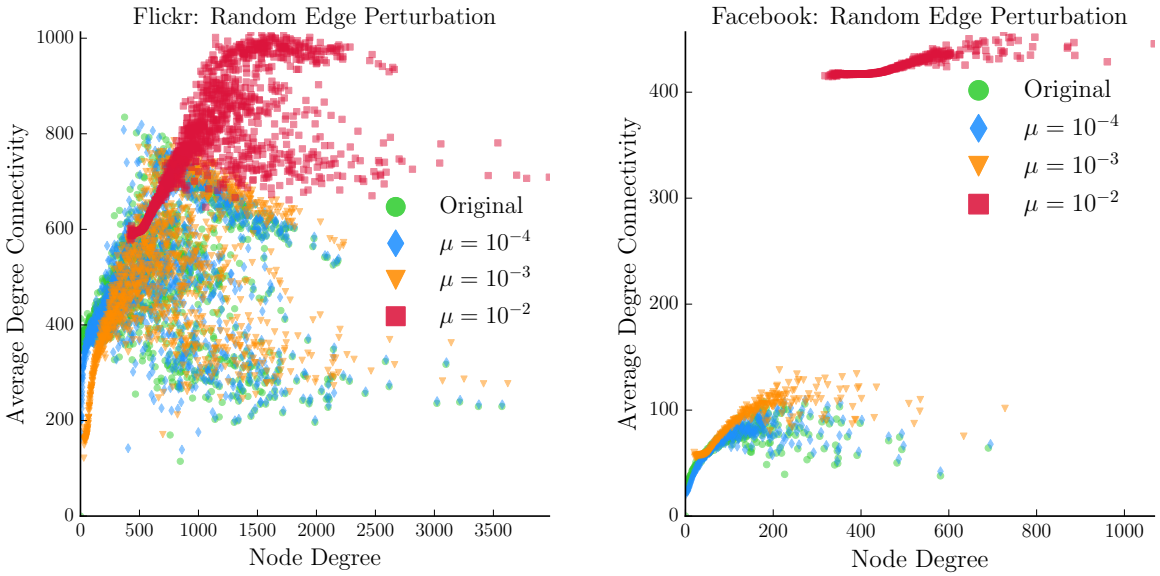


Figure 4.20: REP: Degree Connectivity vs. Node Degree

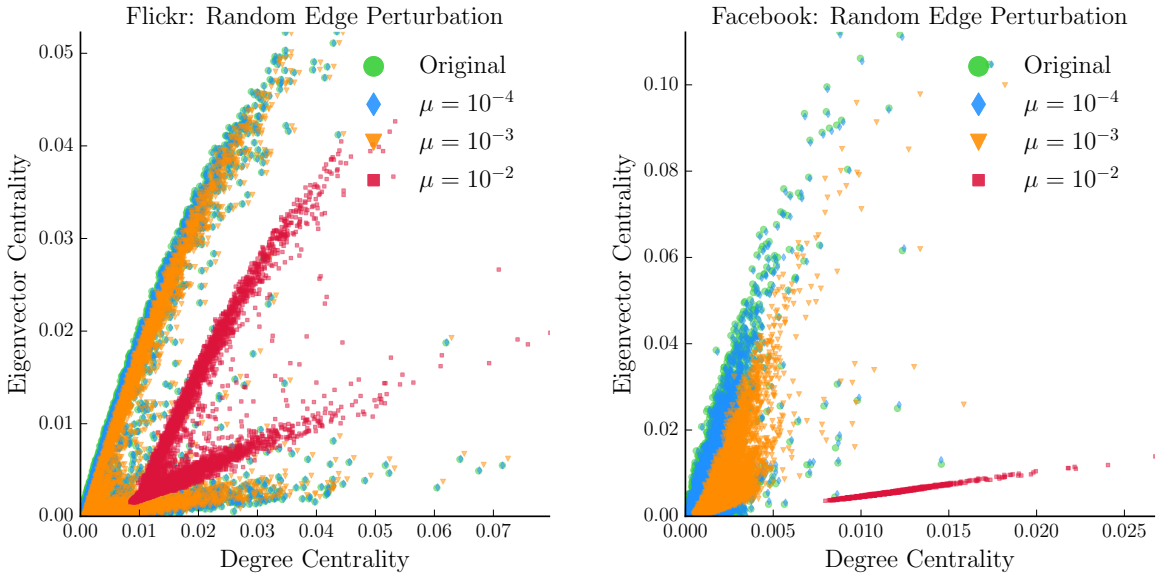


Figure 4.21: REP: Eigenvector Centrality vs. Degree Centrality

- Degree distribution (Figure 4.23) – roughly flatlines for higher-degree nodes as such nodes are fewer in number and the perturbation increases the frequency of each distinct degree to a minimum value.

- Joint degree distribution (Figure 4.24) – forms a checkered pattern for high-degree nodes as the perturbation introduces new edges among high-degree nodes.

- Average degree connectivity (Figure 4.25) – increases sharply for low-degree nodes due to being connected to high degree nodes whose degree has been increased during perturbation.

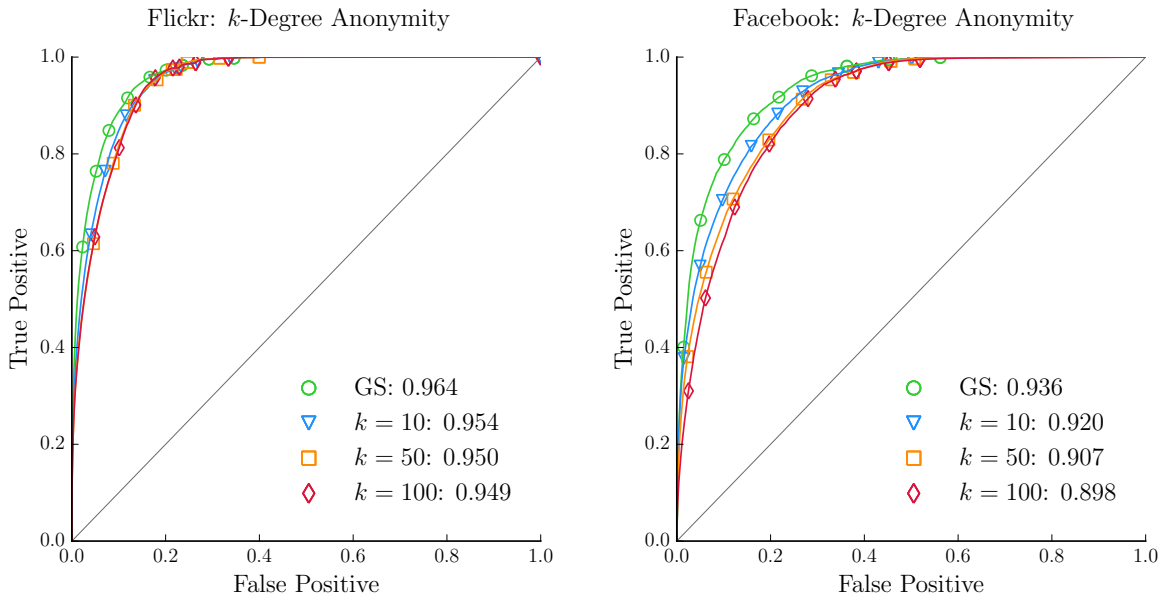


Figure 4.22: KDA: ROC curves

- Eigenvector centrality (Figure 4.26) – remains largely same except for vertical patterns appearing for higher-degree nodes (higher degree centrality) due to perturbation in their degree.

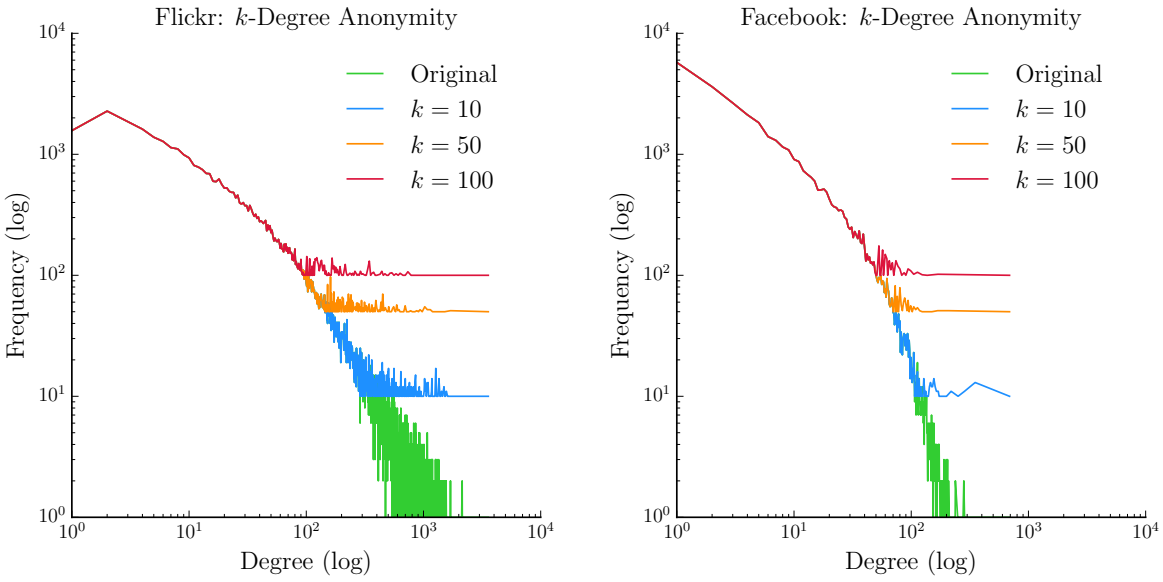


Figure 4.23: KDA: Degree Distribution

4.3.6 1-hop k-Anonymization (1HKA)

Anonymity. 1HKA is inefficient for large graphs with high average node degree. The scheme ensures 1-hop k-anonymity by inserting edges which are of the order of 12% of

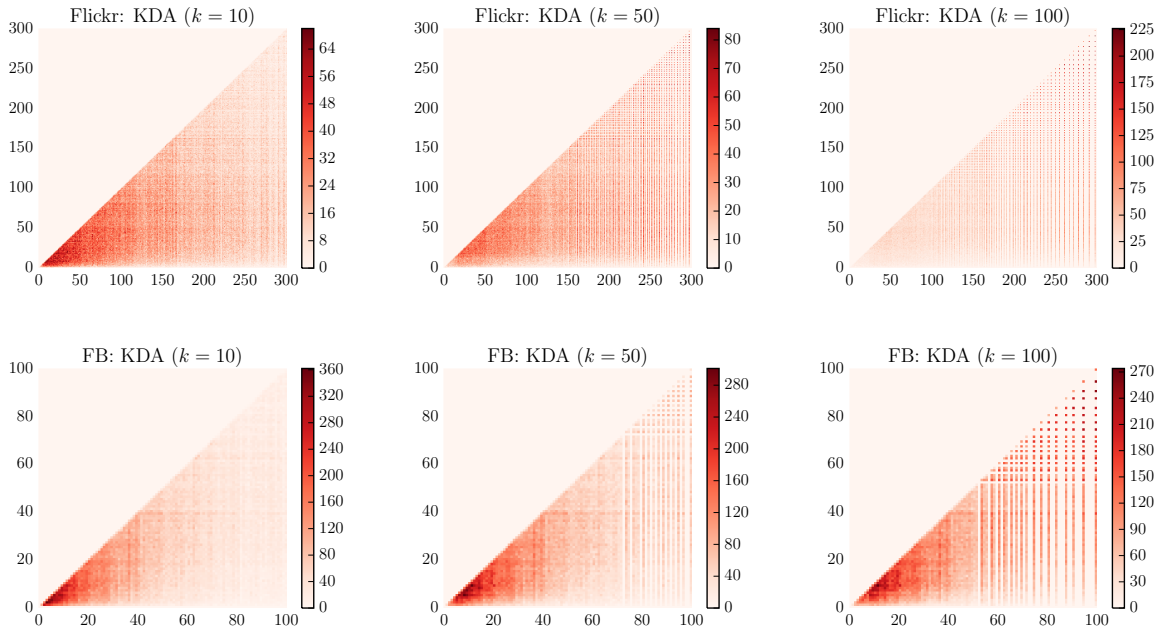


Figure 4.24: KDA: Joint Degree Distribution

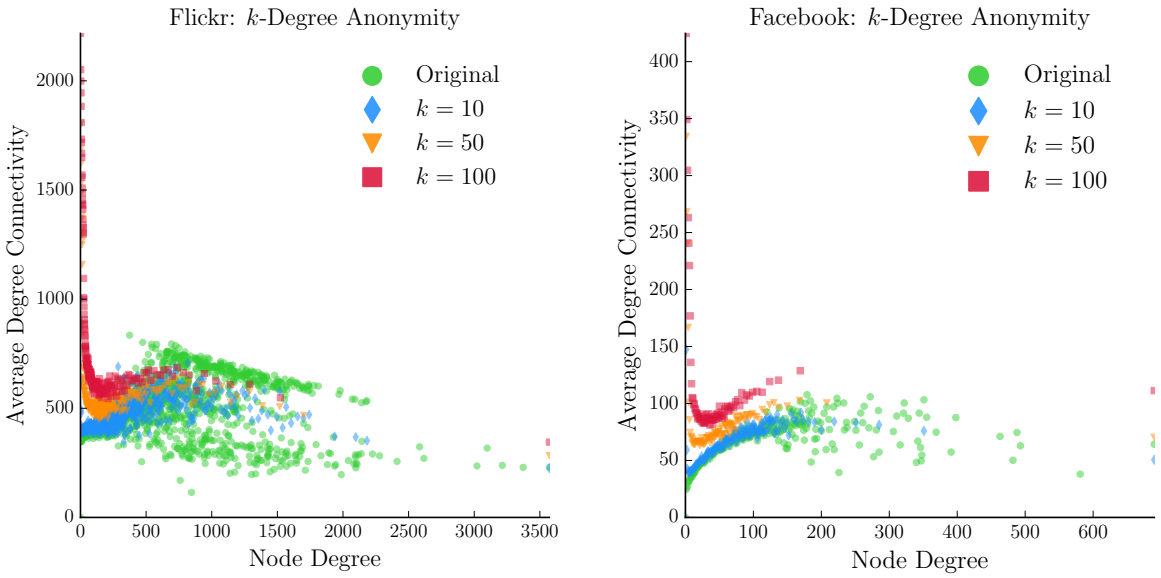


Figure 4.25: KDA: Degree Connectivity vs. Node Degree

the total edges for $k = 30$ [96]. To analyze this scheme and get around the problem of inefficiency we introduce $\mu = (0.10, 0.25, 0.50)$ fraction of edges at random in the graph. Inserting edges at random makes structural de-anonymization harder as compared to inserting them in a formulaic manner (as confirmed by the results of `KDA`), also utility is better preserved in this scenario. Hence the results achieved provide a lower bound on de-anonymizability and an upper bound on utility compared to using the actual scheme. Note that by definition 1-hop features of all nodes in an equivalence class would be completely identical in this scenario therefore they cannot be used to differentiate among node pairs

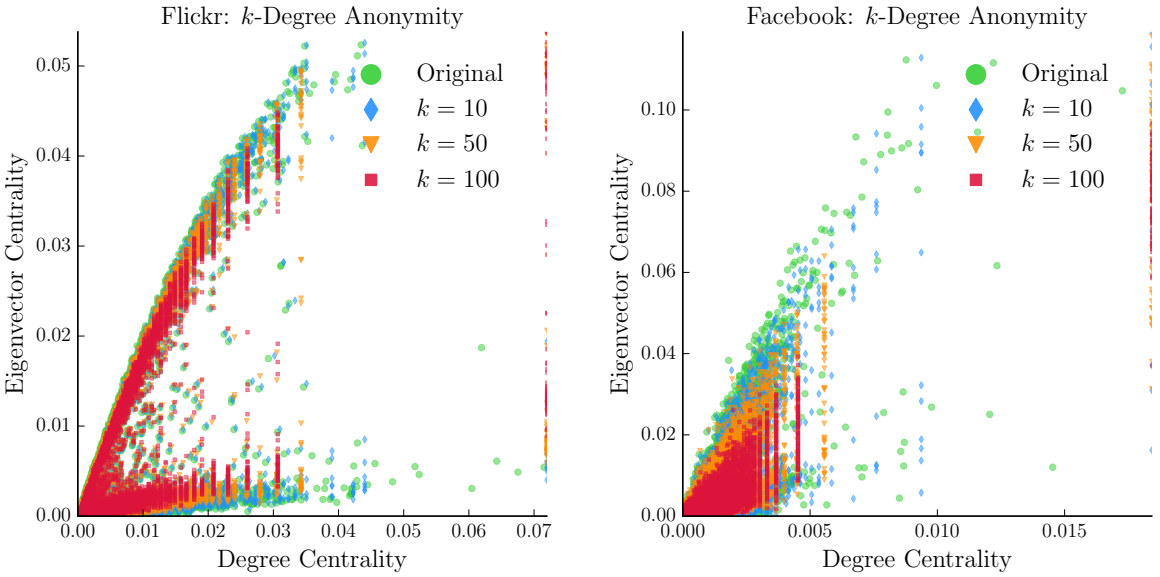


Figure 4.26: KDA: Eigenvector Centrality vs. Degree Centrality

any more. The 2-hop features are still relevant; we replace the 1-hop features by 3-hop features. This is a simple swap which is easily done in our framework thus highlighting its proclivity to swift adaptation. Figure 4.27 confirms that even adding a high fraction of edges does not provide much anonymity.

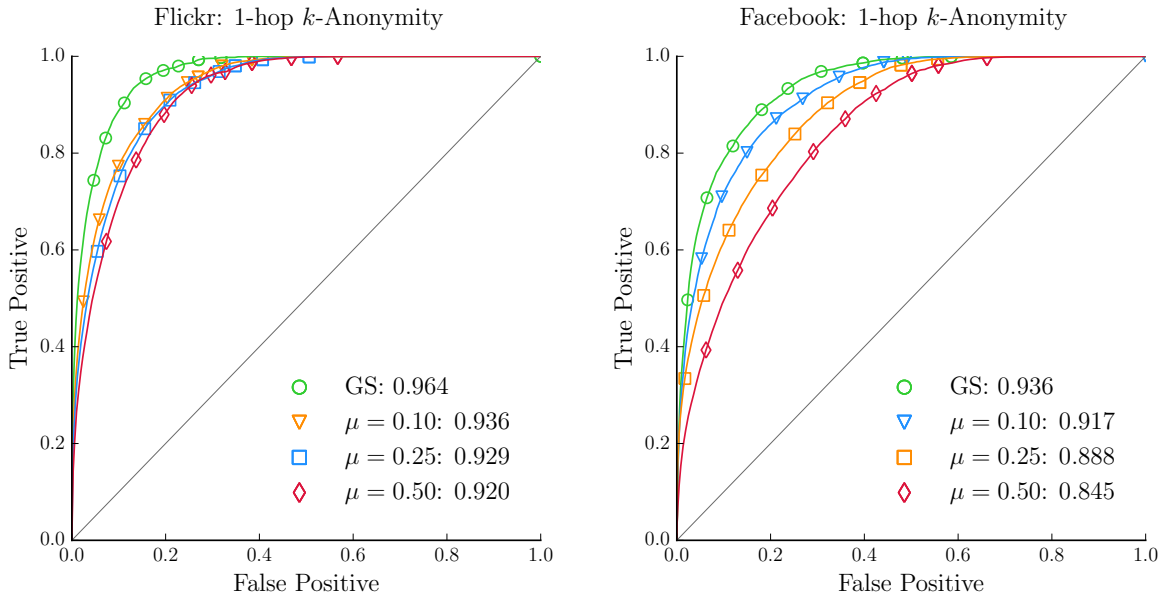


Figure 4.27: 1HKA: ROC curves

Utility. Adding edges at random damages the degree and joint degree distributions but is less harsh on other properties.

- Degree distribution (Figure 4.28) – shift similar to `REP` due to introduction of edges.

- Joint degree distribution (Figure 4.29) – similar to `REP`, shifts diagonally as degree of all the nodes increases. The shift is more concentrated towards the low-degree nodes as most edges are introduced in their vicinity.
- Average degree connectivity (Figure 4.30) – behaves similar to `REP`.
- Eigenvector centrality (Figure 4.31) – behaves similar to `REP`.

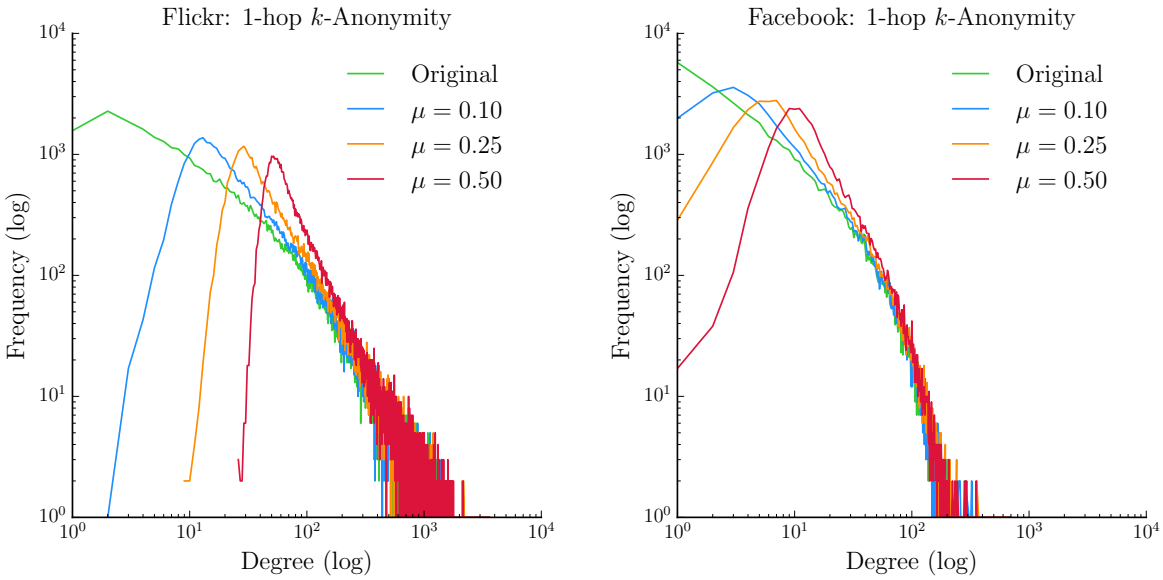


Figure 4.28: 1HKA: Degree Distribution

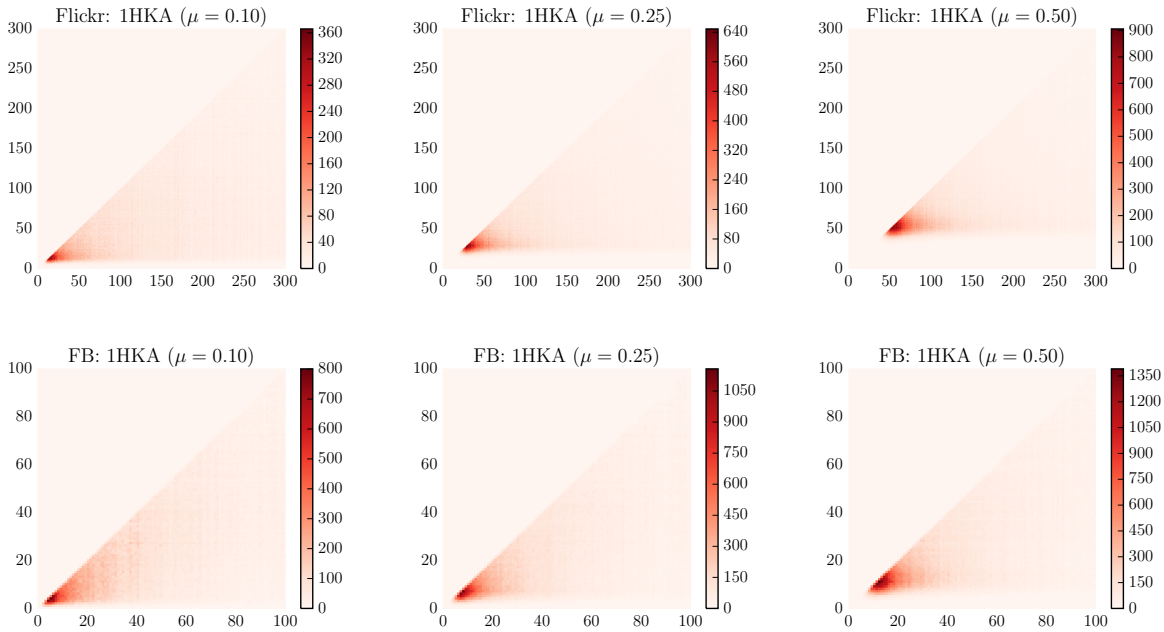


Figure 4.29: 1HKA: Joint Degree Distribution

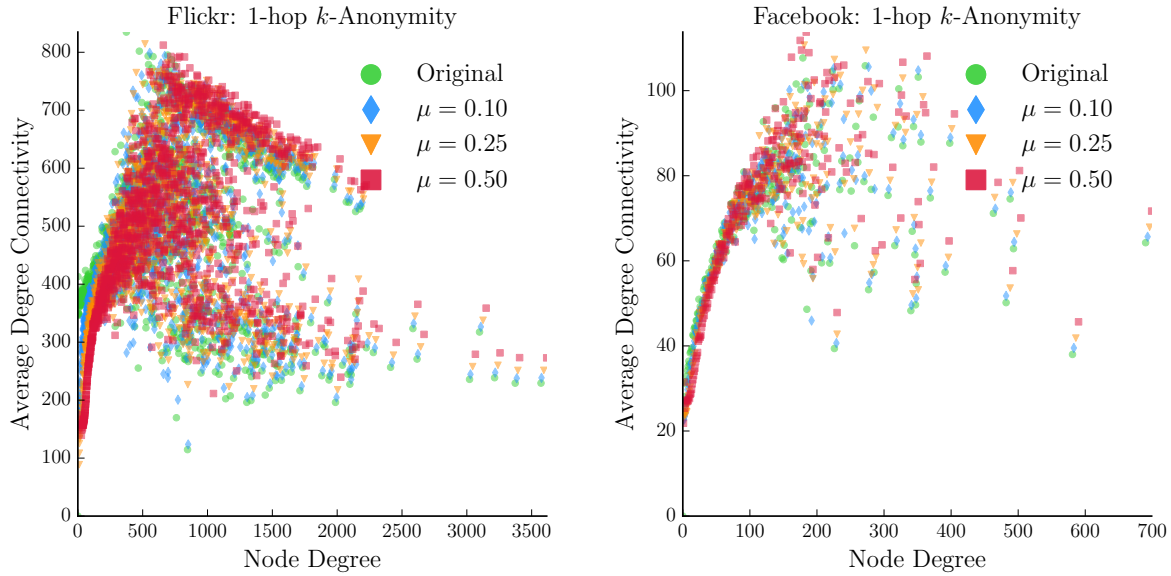


Figure 4.30: 1HKA: Degree Connectivity vs. Node Degree

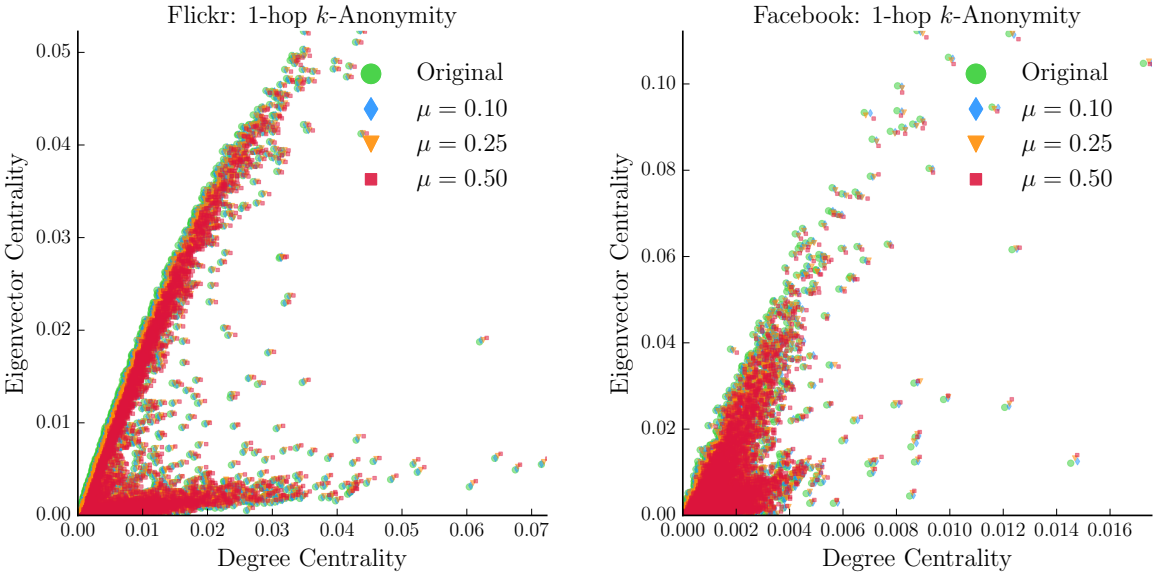


Figure 4.31: 1HKA: Eigenvector Centrality vs. Degree Centrality

4.4 Comparing social graph benchmarking schemes

Even though a multitude of social graph anonymization schemes have been proposed, little work has been done in the area of benchmarking them by quantifying the anonymity and utility they provide.

A-posteriori analysis of re-identification probabilities is a popular [82–84] approach to quantify anonymity in anonymized social graphs. After an adversarial model has been defined such analysis proceeds by formulating queries based on the adversary's a-priori knowledge of the target and then computing the risk of re-identification based on the

induced equivalence relation. This methodology is hard to quantify due to being computationally intensive. The results are highly dependent on the adversarial model and can only handle simple adversaries [82]. Constructing optimal queries for an adversary possessing knowledge of the target's sub-graph is very tricky.

Hay *et al.* [83] present an adversarial model where the adversary knows some structural information about a target in the original graph and tries to identify it by querying the published anonymized graph. The authors assume that all the resulting candidates of the query are equally likely and enforce the condition that each such query should have at least k candidates; this is defined as *K-Candidate Anonymity*. This approach has two serious issues. First, treating all candidates as equally likely is simplistic, undermines the adversary and misrepresents reality. Second, the authors include a candidate in the result of a query only when it exactly matches the structure of the target queried, there is no provision for a probabilistic match, this is not optimal use of resources by the adversary. The authors also present an attack where the adversary explores the neighborhood of the target by performing a breadth first search. This models the scenario where the adversary has incomplete knowledge of the sub-graph around the target. The adversary is limited by the number of edges it can explore; not all edges convey equal information but the adversary does not optimize for this. This attack is more realistic but hard to generalize as it does not describe an adversary's background knowledge concretely. Another serious drawback of the analysis is that the authors compute k-candidate anonymity purely based on the degree of the nodes in the resulting perturbed graph, thus implying that the adversary knows the target degree in the perturbed graph as well. Such analysis befuddles the quantification of anonymity by overestimating the powers of a weak adversary. The authors conclude that anonymization using `RAD` and `RSW` makes such attacks harder, but to provide any meaningful protection the perturbation required is too high to preserve graph properties.

Bonchi *et al.* [82] refine the definition of *K-Candidate Anonymity* by proposing an entropy-based metric. Hay *et al.* [83] measure the probability of a node in the perturbed graph to have originated from the target, thus providing a local estimate. Bonchi *et al.* [82] provide a global estimate by computing the entropy from the distribution of belief probabilities of the nodes being mapped to the target. The authors present two approaches to quantify anonymity – first, where the adversary's aim is to identify a particular target in the anonymized graph and second, where the adversary is interested in identifying anyone in the anonymized graph. The authors compute the entropies for both definitions under `RSP`, modeling a rather weak adversary that only has the knowledge of target's degree. These computations however, become far too complicated and computationally infeasible for even a slightly more potent adversary that has the knowledge of target neighborhoods instead of just the target degree. It is suggested that the adversary also faces a similar challenge while launching attacks, although no motivation is provided for the adversary to do so, nor is it proven that this is the optimal attack. Additionally, this model prevents

us from comparing schemes.

Singh and Zhan [167] propose a metric based on degrees and variance of clustering coefficients of nodes to measure topological anonymity of a social graph. It is not explained why these properties should be picked over others. Additionally, their metric is rather too basic to accommodate adversaries with structural knowledge of the target's neighborhood.

Ji *et al.* [102] propose *SecGraph* – a platform to evaluate graph data anonymization and de-anonymization schemes. The authors conduct a survey in which they analyze a variety of graph anonymization schemes based on utility preservation and their resistance to modern structural graph de-anonymization attacks. The authors conclude that the analyzed anonymity schemes preserve some utility while being vulnerable to most de-anonymization attacks. The adversarial model used to evaluate the de-anonymization attacks is far too liberal towards the adversary. The auxiliary graph used to attack the sanitized graph is sampled from the same graph with varying levels of edge deletion. The resulting pair of graphs have the same set of nodes but the set of edges retained in each graph depends upon the sampling probability s. Specifically, given a graph $G = (V, E)$ with node set V and edge set E, the sampled graph $G' = (V, E')$, where $\Pr[e \in E' \mid e \in E] = s$. The anonymization schemes are evaluated by anonymizing the raw graph before sampling the graphs; this is strictly weaker than sampling the graphs first and then anonymizing them. The adversarial model does not truly reflect a plausible scenario, independent anonymization of graphs is more realistic. Assuming that the adversary has an auxiliary graph of exactly the same individuals as contained in the sanitized graph is a very strong assumption. In reality it is much more likely that graphs have a partial overlap and have perturbations introduced due to a combination of organic and synthetic processes which should be modeled by anonymizing each graph individually. Such an adversarial model overestimates the success of attacks and does not capture the true trade-off between privacy and utility. We take all these factors into consideration in our adversarial model (§ 4.2.1). So, although the paper presents a useful way to compare anonymization schemes, it is far from ideal. Using success of de-anonymization attacks to compare schemes is in particular not ideal: attacks can be optimized to de-anonymize the largest fraction of nodes while sacrificing accuracy or vice-versa. They can also be tuned to be more successful for certain nodes than others. Structure-based attacks are not designed to serve as a measuring tool.

In contrast to such previous approaches, in this chapter we propose a machine-learning framework to benchmark graph anonymization schemes. Our framework can easily adapt to changes in the adversarial model and can accommodate a weak or strong adversary. This is in stark contrast to the rigidity of structure-based de-anonymization attacks which are time consuming to modify and cannot be adapted to a change in adversarial model without manual effort. In the platform presented by Ji *et al.* [102] if an attack readily assumes graph directionality to be available to the attacker [5] then it cannot be easily

applied by an adversary who does not have this information, thus limiting the scope of evaluation. Our framework, being automated, makes changes trivial by just adjusting the node features. None of the attacks previously presented is successful when the anonymized graph is published as a collection of disjoint subgraphs. Availability of seeds would be useless in such a scenario as the mapping would stop at the limit of each subgraph, as shown in Chapter 3 our framework can handle such cases with ease. Most approaches are computationally extensive and thus heavy footed. We provide a nimble and automated alternative to benchmark social graph anonymization schemes.

4.5 Discussion

Anonymizing high-dimensional data is not a new problem, though it keeps resurfacing in one form or another. Data with a large number of attributes are very hard to anonymize without unacceptable information loss [6]. There are exponentially many combinations of dimensions that can be used as quasi-identifiers thus leading to inference attacks. Preventing such attacks requires completely suppressing most of the data which renders publishing data moot. Our experiments confirm the conventional wisdom in the scenario of social graphs.

Parameter choice. The overlap between auxiliary and sanitized graphs is chosen as $\alpha_V = 0.25$ to model an adversary with reasonable side information to launch an attack. A higher α_V strengthens the attacker as the side information increases; however, this does not have any impact on the *relative* success of attacks under different schemes and adversarial models. The de-anonymization attack gives robust results for $\alpha_V \geq 0.20$, we found that results are not reliable for lower values of graph overlap. However, this does not limit the application of our techniques to social graphs of any particular kind. It is challenging to attack sparse graphs due to lack of information but all other attacks face the same challenge. It is possible that in future our techniques could be enhanced to lower the graph overlap needed to launch a potent attack but we believe that there must a minimum amount of side information needed below which attacks are not feasible. This provides further evidence that our attacks should only be considered as a lower limit. Finding the optimum attacker for a given adversarial model is a hard problem, hence it is wise to tread with caution. Other parameters are chosen as detailed in § 3.5.3. We set vector length and bin size (n, b) as (21, 50) (see, § 4.2.3); the value is chosen such that it can accommodate higher degrees and their variation, the choice does not have a huge impact on accuracy (see, § 3.5.3). A forest size of 400 is used to achieve good testing accuracy. We experimented with using features such as centrality, edge weights and group membership in addition to those proposed. Complicated features do not provide a significant improvement.

4.5.1 Risk of re-identification

The classification framework presented quantifies the risk of re-identification based purely on graph structure, given a perturbation scheme. The task of distinguishing identical node pairs from non-identical ones captures the most basic challenge faced by the adversary. We provide a granular graph structure-based metric to capture the likelihood of a node being re-identified. The classification task used by our framework is not new and has been widely used in the literature [3, 5, 117]. The key difference is that previous results have been reported for the global matching task rather than pairwise matching task. The results of the global matching task are derived from the success attained in pairwise matching task. Hay *et al.* [83] proposed *K-Candidate Anonymity* based on the number of matches returned by a structural query on a graph. Bonchi *et al.* [82] refine the definition to propose *k-Preimage Obfuscation* which is based on an entropy. Both these definitions rely on the success of the adversary to find pairwise matchings of graph nodes.

Our structure-based re-identification has a high true positive rate and a low false positive rate, allowing us to re-identify an individual with high confidence. The re-identification rates observed are a lower bound on the attacker's success; more potent structure-based re-identification cannot be ruled out. And structure is not the only information that an adversary might possess; any side information is likely to cause a further decline in false positives. In the light of this, none of the six schemes analyzed provide sufficient anonymity while preserving utility (Figure 4.32). Acceptable levels of anonymity are only achieved at very high perturbation levels at which point data is of little use. Tables 4.1 and 4.2 show that even at very low false positive rates the true positive rates are unacceptably high. `REP` ($\mu = 10^{-2}$) for Facebook is the most successful scheme at repelling structural attacks with true positive value of less than 1% for a false positive of 0.1% and an AUC = 0.585; though not as effective for Flickr it behaves reasonably well. We introduce extreme levels of perturbation in our experiments only as a means to study the graph behavior, perturbation at such high levels serves no practical purpose.

Flickr. All the schemes except `REP` ($\mu = 10^{-2}$) have a TP of above 5% at a FP of 0.1% for even the most extreme level of perturbation. This is already pretty high even if we make the very strong assumption that the attacker can gather no side information for any of the nodes attacked.

Facebook. Anonymization is more successful as compared to Flickr. Most schemes apart from `REP` ($\mu = (10^{-2}, 10^{-3})$) have a TP of around 3–5% at a FP of 0.1%. This also affects the utility of graph metrics which are noticeably damaged. Overall, Facebook's anonymization and utility are more sensitive given a particular level of perturbation.

Table 4.1: Flickr: *False Positive* vs. *True Positive*

False Positive	0.001%	0.01%	0.1%	1%	10%	25%
Graph Split	3.89	9.19	21.07	47.40	88.78	98.56
RSP ($\alpha_E = 0.75$)	1.93	6.41	18.62	44.82	87.36	98.09
RSP ($\alpha_E = 0.50$)	2.59	6.04	15.04	38.85	83.57	97.10
RSP ($\alpha_E = 0.25$)	0.40	1.94	7.48	24.08	76.66	94.71
RAD ($k = 0.10$)	2.21	6.31	16.86	36.38	76.40	94.29
RAD ($k = 0.25$)	1.83	5.05	12.04	27.99	67.62	91.89
RAD ($k = 0.50$)	0.16	0.96	5.27	17.74	51.31	80.50
REP ($\mu = 10^{-4}$)	2.32	7.20	17.53	41.34	85.53	97.39
REP ($\mu = 10^{-3}$)	1.78	4.17	13.63	30.78	71.11	94.11
REP ($\mu = 10^{-2}$)	0.10	0.48	1.92	6.46	27.61	63.95
1HKA ($\mu = 0.10$)	0.11	2.03	14.32	35.66	77.50	94.81
1HKA ($\mu = 0.25$)	0.14	0.93	9.80	31.73	74.54	93.85
1HKA ($\mu = 0.50$)	0.02	0.61	5.59	26.60	70.09	93.58
RSW ($k = 0.20$)	1.05	5.23	15.64	38.11	83.77	99.00
RSW ($k = 0.50$)	0.41	3.09	11.63	29.23	79.04	98.92
RSW ($k = 0.85$)	0.39	2.14	8.44	24.82	79.77	99.22
KDA ($k = 10$)	1.86	6.18	16.70	38.97	84.87	98.58
KDA ($k = 50$)	1.30	5.18	14.54	34.89	81.77	99.04
KDA ($k = 100$)	2.07	5.41	14.87	34.83	80.90	98.64

4.5.2 Comparing anonymization schemes

Bearing in mind that none of the schemes analyzed here are really fit for the purpose, we provide a coarse ranking based on utility preservation and relative anonymity. `RSW` and `KDA` are by far the least useful schemes for anonymizing social graphs. Neither provides a safe level of anonymity even after hugely damaging the graph features. `RSP` and `RAD` are by far the best schemes among the ones that we have analyzed; both provide graceful and gradual degradation of graph utility. `REP` and `1HKA` lie in the middle. `REP` provides

Table 4.2: Facebook: *False Positive* vs. *True Positive*

False Positive	0.001%	0.01%	0.1%	1%	10%	25%
Graph Split	0.32	1.04	4.40	34.32	78.60	94.30
RSP ($\alpha_E = 0.75$)	0.46	3.14	11.47	34.20	75.51	92.16
RSP ($\alpha_E = 0.50$)	0.32	1.71	7.23	25.24	68.17	87.77
RSP ($\alpha_E = 0.25$)	0.13	0.44	2.97	13.69	54.02	77.84
RAD ($k = 0.10$)	0.42	2.49	10.81	32.48	71.61	89.92
RAD ($k = 0.25$)	0.22	0.99	5.16	18.78	56.42	81.00
RAD ($k = 0.50$)	0.11	0.56	2.77	9.78	33.35	58.63
REP ($\mu = 10^{-4}$)	0.17	2.61	10.98	28.21	65.89	86.69
REP ($\mu = 10^{-3}$)	0.20	0.67	2.37	8.19	32.28	58.21
REP ($\mu = 10^{-2}$)	0.00	0.71	0.71	4.44	13.21	30.87
1HKA ($\mu = 0.10$)	0.19	1.87	9.32	30.14	71.82	89.87
1HKA ($\mu = 0.25$)	0.25	3.11	11.13	27.83	61.61	83.79
1HKA ($\mu = 0.50$)	0.07	0.94	4.64	16.86	49.52	75.04
RSW ($k = 0.20$)	0.21	1.38	7.33	23.97	64.93	88.68
RSW ($k = 0.50$)	0.19	1.18	4.19	17.10	57.25	87.21
RSW ($k = 0.85$)	0.00	0.12	1.51	11.17	51.83	86.91
KDA ($k = 10$)	0.58	3.02	11.20	32.33	71.08	91.49
KDA ($k = 50$)	0.21	1.49	8.01	26.55	66.26	89.30
KDA ($k = 100$)	0.08	0.91	5.26	19.09	62.15	88.37

anonymity at high levels of perturbation but produces graphs that are a mere shadow of the original ones; it also introduces perturbations proportionate to the number of non-edges which has a very damaging effect on sparse graphs. `1HKA` preserves some properties but is rather damaging for the others. The analysis of `1HKA` provides a lower bound on de-anonymizability and an upper bound on the utility (see, § 4.3.6). Schemes that provide k-anonymity by homogenizing the i-hop neighborhood around nodes are not only computationally expensive but also defenseless against sub-graph attacks. As shown these schemes are very destructive to graph properties and can be defeated by extending the

features beyond i hops. Such schemes achieve complete structural similarity only after either all nodes are fully connected or disconnected [102].

Table 4.3 provides the concrete relation between the deviation of degree distribution and joint degree distribution from the original as characterized by the Hellinger distance and AUC of the ROC curves obtained. The Hellinger distance is defined as the statistical distance $H(P,Q)$ between two discrete probability distributions $P = (p_1, \ldots, p_k)$ and $Q = (q_1, \ldots, q_k)$ given by:

$$H(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{k}(\sqrt{p_i} - \sqrt{q_i})^2}$$

$H(P,Q)$ takes a value in the range [0,1]; the schemes that show a lesser distance between original and perturbed distributions preserve the distribution better. In general schemes which produce distributions closer to the original ones tend to allow more successful structural attacks as measured by the AUC. Figure 4.32 shows how various anonymization schemes fare against each other for a *reasonably* chosen level of perturbation.

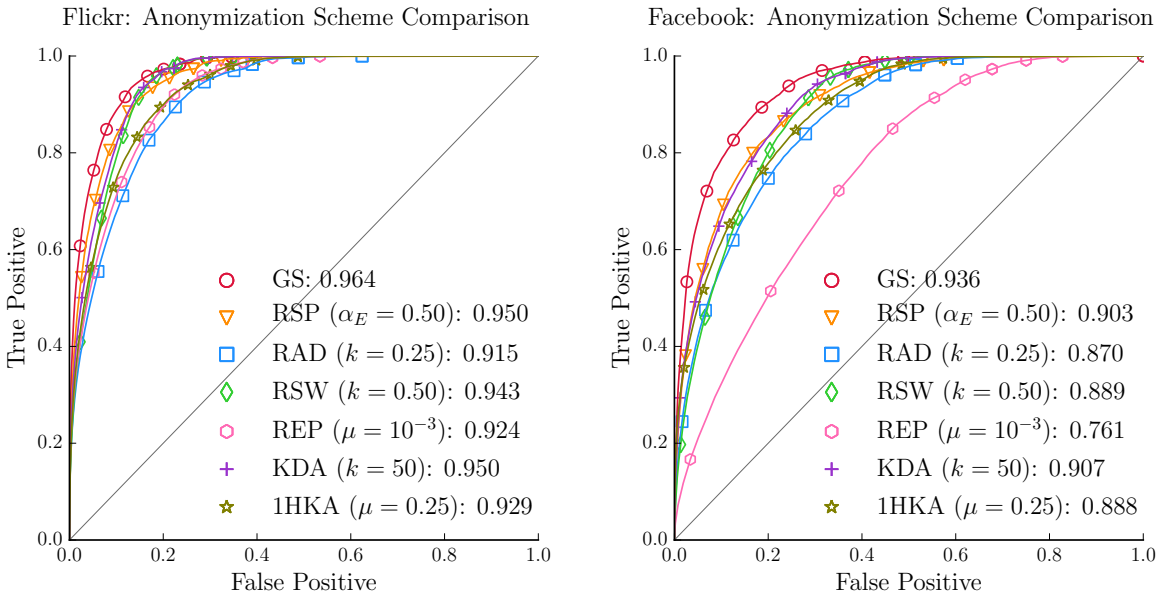


Figure 4.32: Scheme Comparison: ROC curves

Bottom line. After carefully studying the results we do not believe any scheme can guarantee anonymity while preserving utility. De-anonymization can be seen as a utility metric as it is constructed from the properties of the graph and it cannot be damaged without destroying other properties. In summary:

- Our experiments show that properties of dense graphs are more resilient to a proportionate perturbation which in turn makes them more vulnerable to attacks. De-anoymization of Flickr is less sensitive to edge perturbations than Facebook as dense

Table 4.3: Hellinger Distance between degree distribution (DD) and joint degree distribution (JDD) for perturbed and unperturbed graphs and its effect on ROC curve's Area Under the Curve (AUC)

	Flickr			Facebook		
	DD	JDD	AUC	DD	JDD	AUC
RSP ($\alpha_E = 0.75$)	0.109	0.570	0.959	0.062	0.295	0.926
RSP ($\alpha_E = 0.50$)	0.130	0.567	0.950	0.100	0.340	0.903
RSP ($\alpha_E = 0.25$)	0.204	0.610	0.931	0.194	0.477	0.850
RAD ($k = 0.10$)	0.314	0.562	0.935	0.138	0.283	0.917
RAD ($k = 0.25$)	0.467	0.582	0.915	0.273	0.336	0.870
RAD ($k = 0.50$)	0.603	0.657	0.864	0.439	0.478	0.763
REP ($\mu = 10^{-4}$)	0.232	0.568	0.955	0.280	0.286	0.900
REP ($\mu = 10^{-3}$)	0.599	0.612	0.924	0.759	0.630	0.761
REP ($\mu = 10^{-2}$)	0.912	0.899	0.792	0.999	1.000	0.585
1HKA ($\mu = 0.10$)	0.318	0.562	0.936	0.141	0.269	0.917
1HKA ($\mu = 0.25$)	0.465	0.581	0.929	0.281	0.293	0.888
1HKA ($\mu = 0.50$)	0.584	0.606	0.920	0.428	0.338	0.845
RSW ($k = 0.20$)	0.000	0.170	0.953	0.000	0.084	0.904
RSW ($k = 0.50$)	0.000	0.268	0.943	0.000	0.143	0.889
RSW ($k = 0.85$)	0.000	0.350	0.942	0.000	0.203	0.879
KDA ($k = 10$)	0.136	0.576	0.954	0.056	0.279	0.920
KDA ($k = 50$)	0.260	0.793	0.950	0.125	0.485	0.907
KDA ($k = 100$)	0.327	0.863	0.949	0.175	0.603	0.898

graphs retain enough information even after perturbation. Additionally, high-degree nodes are more vulnerable than low-degree ones. All useful anonymization schemes leave a certain fraction of true edges intact; only a few *true* friends are needed to re-identify an individual. Thus dense graphs are more vulnerable to re-identification attacks than sparse graphs since on an average each node has more friends.

- Deleting edges is less harmful than adding false edges; Bonchi *et al.* [82] made the

same observation. Introducing random edges disrupts the small-world characteristics of the graph by shrinking it, while removing edges at random still leaves paths that preserve the small-world features.

- Increasing perturbation does not necessarily provide more anonymity in all cases but it always degrades graph utility.
- Formulaic and local graph perturbation such as `KDA` fares worse at providing anonymity than global graph perturbation. The interconnectivity of graphs allows leveraging the unperturbed neighborhoods to attack the perturbed neighborhoods. An anonymization scheme must be global to have any chance of providing privacy.
- The discovery of more potent structure-based re-identification would not alter the relative benchmarking of the schemes. Structure-based re-identification which is generalizable would always be more successful on a weaker scheme. The classification framework presented by us is generic as its success increases with decrease in strength of anonymization for a particular scheme. This claim is supported by the results presented for six different schemes.

4.6 Summary

In this chapter we extended and built upon the machine-learning techniques presented in Chapter 3 to construct a benchmarking platform for perturbation-based social graph anonymization schemes. The platform allows us to evaluate and compare anonymization schemes efficiently. The work bridges the fundamental research gap of comparing and evaluating social graph anonymization schemes under a common framework. We overcome a major hindrance to learning based attacks on overlapping graphs by proposing a mechanism to train the learning model in absence of ground truth or seed mappings. We conduct a thorough analysis of the effect of graph perturbation on the anonymity achieved and utility preserved by studying six perturbation-based social graph anonymization schemes using publicly-available real-world social graphs. We comprehensively analyze the effect of graph perturbation on utility by studying five fundamental graph metrics. We examine whether the anonymization schemes provide any anonymity at all, even when perturbation levels are high enough to destroy almost all utility. We propose a granular graph-structure-based metric to capture the likelihood of node re-identification. The schemes are compared based on the success of structure-based re-identification with varying levels of graph perturbation and its relation to graph utility. The findings are supported by a thorough and comprehensive analysis of graph-anonymization schemes using real-world social networks.

In Chapter 5 we will demonstrate how the success of our classifier lies at the heart of social graph de-anonymization by recovering actual mappings across anonymized graphs.

Chapter 5

The next generation of social graph de-anonymization attacks

5.1 Introduction

As discussed in previous chapters, in the event of a data release the privacy of individuals is often seen as a hindrance and protected as an afterthought. This manifests itself as poorly thought through data anonymization schemes. Social networks are specially hard to anonymize due to the interconnectivity of individuals. As seen in § 2.4, privacy risks can be alleviated to some extent by interactive mechanisms like differential privacy, where instead of releasing the data a query interface is provided and the replies are tightly controlled by adding noise to provide a privacy guarantee. Such mechanisms have their limitations as they are non-trivial to set up, expensive to maintain and introduce excessive noise where the data analyst's goals are unknown [68]. Non-interactive privacy mechanisms (see, §§ 2.5 and 2.6) that release a perturbed version of the data have also been widely proposed [77–90, 92–98, 168, 169]. As seen in § 2.8, privacy researchers routinely break such mechanisms [4, 5, 36, 87, 103, 104, 106, 107, 114–119, 170, 171]. Overwhelming evidence suggests either privacy or utility must be relinquished if high-dimensional datasets are to be released.

Ji *et al.* [104] show that most anonymized social network datasets can be largely de-anonymized purely based on structural knowledge but searching for the optimal attack is computationally infeasible. None of the approaches so far show performance close to what is possible [104]. All approaches rely on hand-picked heuristics to exploit structural similarity in social graphs. In this chapter we overcome these hurdles by proposing a fresh approach using machine-learning models to obtain an appreciable improvement. In particular, we present a new generation of graph de-anonymization algorithm that is seedless, and automatically discovers artefacts from data samples to construct a de-anonymization learning model thus doing away with the need for hand-selected features.

Learning from pertinent examples limits human intervention and creates models that are robust, reliable, nimble and able to adapt to changes in anonymization strategy.

Our contributions. In this chapter we make the following primary contributions:

- We present a new generation of heuristic-free seedless graph de-anonymization algorithm that uses machine learning to map nodes across graphs (§ 5.3.3).
- We show that our proposed scheme can handle a variety of adversarial models and is agnostic to the de-anonymization scheme employed (§ 5.3.4).
- The algorithm is evaluated on three real-world social graph datasets under four adversarial models (§ 5.4).
- We conduct a thorough comparison of our algorithm with seven seed-based and seedless attacks using two real-world social network datasets. Our algorithm's overall performance is better than all the others by a healthy margin (§ 5.5).

The work presented in this chapter is based on the paper titled *Change of Guard: The Next Generation of Social Graph De-anonymization Attacks* [172] published in the Proceedings of the 9th ACM Workshop on Artificial Intelligence and Security (AISec 2016).

5.2 The de-anonymization landscape

Graph de-anonymization traditionally tries to obtain a mapping between two graphs using just their structure. The graph nodes represent individuals and the edges represent relations among them. After an anonymized graph is released the adversary tries to match it with a known graph to compromise privacy. The adversary's knowledge is often imperfect, but may be sufficient to launch potent re-identification attacks [3, 5]. Identification of individuals present in both graphs could lead to discovery of sensitive relations among them. The adversary could obtain side information from a number of sources such as another data release, scraping the web or by colluding with individuals who are part of a common social network. Most data releases aim to preserve some utility to facilitate analysis thus limiting possible perturbation of the data. This allows the attackers to try unraveling the anonymization using structural similarity and if available, any other side information.

5.2.1 Anonymizing social networks

Several social network anonymization schemes try to protect node privacy, i.e., concealing an individual's presence in the graph (see, §§ 2.5 and 2.6). Such schemes are mainly

of two types [43–45] – clustering-based and perturbation-based. Clustering-based graph anonymization schemes release aggregate graph information instead of the raw graph while perturbation-based schemes introduce imperfections in the graph via a combination of edge additions/deletions to deter graph-structure-based de-anonymization attacks. Summarizing data limits its scope and usage but provides better privacy. This chapter considers perturbation-based schemes, which are more popular due to their wider applicability and simplicity.

5.2.2 Heuristics vs. machine learning

Perfect recovery of common nodes across two graphs is a hard problem and computationally infeasible for large graphs in general [127]. The problem becomes even harder when the two graphs are noisy and the edges are perturbed. To make the problem tractable an imperfect and incomplete mapping is computed whose goal is to minimize the error and maximize the mappings. Modern graph de-anonymization algorithms are quite potent and can re-identify a large fraction of nodes with good accuracy [4, 5, 36, 87, 103, 104, 106, 107, 114–119]. All such algorithms exploit graph heuristics to recover nodes based on structural properties. This approach, although potent, cannot produce the best performance with changing real-world anonymization schemes. Additionally there is no predefined manual to choose heuristics. The choice is based on judgment and prior knowledge of anonymization techniques and is a constant co-evolution of attack and defense.

As demonstrated in Chapters 3 and 4, basic features based on neighborhood degree can be used to build complex and expressive models that capture the learning task very precisely [159]. Handcrafting heuristics also suffers from the problem of tuning the parameters for various adversarial models, datasets and graph metrics which is done by trial and error. Pedarsani *et al.*'s [118] approach is a prime example of the limitations. The authors present a Bayesian model to capture the similarity of two individuals belonging to different graphs. The model requires knowledge of the anonymization scheme and can only handle simple features as it gets too complicated when feature complexity rises. Machine learning models can handle such complications and do not require manual tuning. In the rest of chapter we show how to design such systems to breach privacy.

5.3 Evaluation

In this section we describe the adversarial model under which our de-anonymization algorithm is evaluated. We also define the key components of the learning model and their role in deconstructing the anonymization of social graphs.

5.3.1 The adversarial model

We evaluate our attack using the model outlined in § 2.7. The adversary uses auxiliary graph G_{aux} as side information to compromise identities in the sanitized graph G_{san}.

5.3.2 Learning to de-anonymize

As discussed in earlier chapters, we use a random-forests machine learning model to de-anonymize social graphs. The learning task is to classify a pair of nodes selected at random from G_{aux} and G_{san} as identical or non-identical. In this section we lay the foundation of the model and show how to construct it. The simple learning task of node pair classification lies at the heart of our algorithm. We show how success in the learning task translates to success at de-anonymization.

Features

We use the same basic principle as described in §§ 3.3.3 and 4.2.3 to define a node feature vector. The neighbors of a graph node can be split into two categories, 1-hop nodes and 2-hop nodes. Node feature vectors can be created for both 1-hop and 2-hop neighbors of a node in an undirected graph – a total of two node categories. In a directed graph the 1-hop nodes are divided into successors and predecessors while the 2-hop nodes can be divided into successor-of-successor, successor-of-predecessor, predecessor-of-successor and predecessor-of-predecessor – a total of six node categories. A node feature vector is defined as a concatenation of the two degree distribution vectors for undirected graphs and 12 degree distribution vectors for directed graphs, counting both in-degree and out-degree for the six node categories. Figure 5.1 shows a sample feature vector composed of quantized neighborhood degree distribution for a particular node category.

Additionally, we calculate the Silhouette Coefficient of the number of 1-hop neighbors as well as 2-hop neighbors for a node pair to decide the split in a tree node. The Silhouette Coefficient of a node pair is defined as:

$$\delta(d_1, d_2) = \frac{\mid d_1 - d_2 \mid}{\max(d_1, d_2)}$$

where d_1 and d_2 are the number of either 1-hop neighbors or 2-hop neighbors of nodes belonging to G_{aux} and G_{san} respectively. This trains the model to predict the degree deviation for identical and non-identical node pairs. Modularity of features makes them nimble and adaptable; the features represent a node pair which is a data point for the learning model.

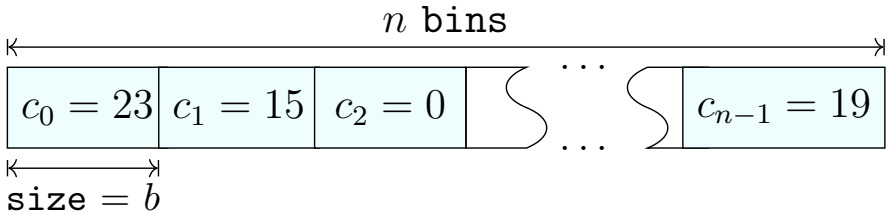


Figure 5.1: Feature vector of quantized node neighborhood

Training

The model is trained as discussed in § 4.2.3, by sampling training data by splitting G_{aux} and G_{san}. This can be achieved by the adversary with just a rough estimate of the node set overlap between the two graphs.

Classification

After training the model, an unseen node pair can be classified as described in § 3.3.4. Classification proceeds by extracting features of the nodes in the pair and passing it through the random forest. Passing the node pair through a tree assigns it a probability of being identical. After the node pair has been through all the trees, the accumulated probabilities are averaged thus assigning the node pair a final prediction score which is a real number in [0,1]. The higher the score, the more likely is the node pair to be identical.

5.3.3 Unraveling anonymization

Classifying node pairs provides a measure of their structural similarity but it cannot be used directly to identify true mappings. Choosing a high classification threshold can be used to select identical node pairs with high likelihood. However, false positives cannot be completely avoided; too high a threshold will rule out a number of node pairs which in spite of being identical, will not be selected thus adversely affecting the number of mappings identified. The problem can be solved by two key observations [5]; first, high degree node pairs are easier to re-identify than low degree node pairs due to their higher information content and second, common friends of a node pair provide a valuable metric for identification [150]. The rest of the section describes how these observations can be used along with structural similarity as quantified by the classifier to produce actual node mappings.

3-phase re-identification. We carry out re-identification of node pairs in phases, the key idea is to re-identify high degree nodes early and then use them to attack low degree nodes. Since high degree nodes are easier to attack we divide the node pairs into three

categories based on their degree. We pick three degree thresholds such that $t_1 > t_2 > t_3$ and divide the node pairs. All pairs with the degree of both nodes greater than t_1 fall into *Phase 1*, all pairs with degree of both nodes greater than t_2 but not a part of *Phase 1* fall into *Phase 2* and finally all pairs with degree of both nodes greater than t_3 but not a part of either *Phase 1* or *Phase 2* fall into *Phase 3*. Nodes of degree lower than or equal to t_3 are left alone; t_3 is chosen as a low value and such nodes are hard to identify with high accuracy. Moreover, they are not influential and evoke little interest in an adversary. We train a separate random forest for each phase to focus the attack better. The attack begins with Phase 1 node pairs, moves on to Phase 2 node pairs and concludes with Phase 3 node pairs.

Initial mappings. To produce the initial set of mappings all node pairs in Phase 1 are classified and only pairs with classification score above 0.95 are selected. These mappings are *dirty* since at this stage a node might appear in multiple mappings thus making them contradictory.

Cleaning mappings. A set of dirty mappings is *cleaned* by greedily selecting node pairs starting with the node pair with the highest classification score. After a node pair is selected all further instances of both the nodes in the pair are discarded from the remaining mappings. Cleaning improves the overall accuracy of the mappings.

Filtering node pairs. The identified initial mappings are treated as true. At this stage, the mappings are bound to have errors but we have no way to differentiate between a true mapping and a false one. However, the initial mappings are composed of node pairs which are structurally very similar, even the non-identical pairs, the nodes are highly likely to be in close proximity of each other. We leverage this knowledge to filter node pairs based on the friends they share. All node pairs (including those present in the initial mappings) are filtered based on the cosine similarity of their neighbors that have been identified in initial mappings. Cosine similarity is defined as:

$$CS(X, Y) = \frac{|X \cap Y|}{\sqrt{|X| \cdot |Y|}}$$

where X and Y are two non-empty sets. The filtering process only preserves node pairs that have a cosine similarity above a threshold. For a directed graph the cosine similarity is calculated as a sum of cosine similarity of common successors and predecessors. The filtered node pairs are cleaned again to discover new mappings; however, this time the classification threshold is relaxed and node pairs self-select themselves until all pairs are exhausted. The new mappings discovered at the end of this step are preserved and previous mappings are forgotten.

Propagating the mappings. Filtering node pairs using the initial mappings increases the accuracy and coverage of subsequent mappings. Performance is further boosted by iterating the process. As new mappings are discovered, filtering node pairs become more reliable due to increase in accuracy and number of common friends; additionally, a larger set of mappings reaches higher number of nodes. This produces a snowball effect causing accuracy and coverage to grow till they stabilize. After this process concludes for Phase 1 the mappings are frozen and we continue with Phase 2. As we decrease the degree threshold to t_2 the number of node pairs rise sharply, making it inefficient to classify all node pairs. This is where the mappings from Phase 1 come in handy; the Phase 2 node pairs are pre-filtered before classification. Only the filtered mappings are classified and cleaned; additionally, we decrease the number of node pairs further by removing the nodes that have already been mapped in Phase 1. In Phase 2 node pairs are first filtered using Phase 1 mappings, after new mappings are discovered they are rolled in with the frozen Phase 1 mappings and the process is repeated. Whenever new mappings are found at a later phase they are rolled in with the frozen mappings, keeping only the latest mappings from the current phase. After the mappings from Phase 2 stabilize they are frozen and we repeat the same process for Phase 3, starting with filtering using frozen Phase 1 and Phase 2 mappings. We observe that mappings stabilize a lot quicker in later phases, as after enough mappings have been discovered, new mappings do not influence the filtering much.

5.3.4 Datasets and implementation

The proposed 3-phase seedless attack (`3PSL`) is evaluated using three publicly available real-world social networks, Flickr [161] – a popular social network for sharing pictures, Epinions[1] – a who-trust-whom online social network of a general consumer review site[2] and Enron[3] – a email communication network. The Flickr graph is undirected and provides group membership of the users, Epinions is directed and has no group information and Enron is an undirected graph. The number of nodes and edges in the original graphs are as follows: Flickr (nodes = 80 513, edges = 5 899 882), Epinions (nodes = 75 879, edges = 508 837) and Enron (nodes = 36 692, edges = 183 831). These graphs are used to produce overlapping auxiliary and sanitized graphs with varying node and edge overlaps (see, § 5.3.1); the details are summarized in Table 5.1. The number of common nodes in each phase is depicted in Table 5.2 and Table 5.3 shows the chosen degree thresholds (see, § 5.3.3).

We analyze our attack with four different adversaries:

(*i*) $\mathcal{A}_1$ – Flickr undirected graph and node group membership ($\alpha_E = \alpha_V = 0.33$);

[1] `https://snap.stanford.edu/data/index.html`

[2] `http://epinions.com`

[3] Same URL as Footnote 1

(*ii*) $\mathcal{A}_2$ – Epinions directed graph ($\alpha_E = 0.33, \alpha_V = 0.20$);

(*iii*) $\mathcal{A}_3$ – Epinions directed graph ($\alpha_E = 0.50, \alpha_V = 0.35$);

(*iv*) $\mathcal{A}_4$ – Enron undirected graph ($\alpha_E = 0.43, \alpha_V = 1$).

This allows us to test our structural attack under varying graph densities, edge overlaps, node overlaps and side information like directionality and group membership. A Flickr node pair is considered identical only if the group membership matches exactly. Table 5.4 shows the number of samples for the three phases used to train the random forest classifier.

Table 5.1: Graph details

	G_{aux}		G_{san}	
	Nodes	Edges	Nodes	Edges
Flickr ($\alpha_E = \alpha_V = 0.33$)	51 594	1 322 970	51 698	1 302 064
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	30 832	114 908	30 584	113 461
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	41 651	183 284	41 673	193 830
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	36 692	110 298	36 692	110 298

Table 5.2: Common nodes

	Phase 1	Phase 2	Phase 3	Total
Flickr ($\alpha_E = \alpha_V = 0.33$)	8684	6608	2779	18 071
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	759	1234	783	2776
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	1008	2727	1603	5338
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	1462	2214	8357	12 033

Table 5.3: Degree thresholds for all phases

	t_1	t_2	t_3
Flickr ($\alpha_E = \alpha_V = 0.33$)	30	9	5
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	30	9	5
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	50	9	5
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	25	9	2

Table 5.4: Number of identical (**I**) and non-identical (**NI**) samples used for training

	Phase 1		Phase 2		Phase 3	
	I	**NI**	**I**	**NI**	**I**	**NI**
Flickr ($\alpha_E = \alpha_V = 0.33$)	8715	1 452 188	8765	1 988 697	4002	1 994 579
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	658	1 010 758	1303	1 998 397	920	1 998 940
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	1110	1 595 258	3729	1 996 012	2063	1 997 804
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	2219	3 338 583	3520	1 996 121	14 364	1 985 529

5.4 Results

In this section we present the analysis of results obtained by running `3PSL` on the three datasets and four adversarial models. The classification of node pairs is successful with consistently high true positive and low false positive rate. This is critical for the filtering to begin as it depends on the quality of initial node mappings. Additionally the classification produces good quality results overall as summarized by the area under the Receiver Operating Characteristic (ROC) curve (AUC). The ROC illustrates how close the classifier is to an ideal one. It does so by measuring the True Positive (TP) rate as the False Positive (FP) rate tolerated is varied in the range [0, 1]. An ideal classifier gives a TP rate of 1 at FP rate 0, whereas TP and FP are always the same for random guessing. In practice a classifier will always make errors (FP), our goal is to maximize the correct classification rate (TP) for the error tolerated. The Area Under the Curve (AUC) provides a summary of the quality of the classifier, an ideal classifier has an AUC = 1 where as random guessing produces a classifier with an AUC = 0.5. Classification success is an important factor behind high quality clean node pairs. Figure 5.2 shows the classification success and Table 5.5 provides a snapshot of the relation between TP and FP rates for Phase 1 for all adversaries. Even at a FP rate of 0.1% the TP rate is about 15-30% which allows us to mount an attack. Table 5.6 lists the number of mappings produced at various phases by `3PSL` as well as the total mappings. The sparser the graph the fewer high degree nodes it has and consequently it has fewer candidates for Phase 1, this in turn affects its propensity to be attacked. On the flip side sparse graphs limit the scope of analysis.

5.4.1 Mapping accuracy and coverage

We use accuracy and coverage to study the performance of the de-anonymization algorithm. Accuracy and coverage are defined as $\frac{m_{true}}{m_{out}}$ and $\frac{m_{true}}{m_{tot}}$ respectively; where m_{out} is the number of mappings output by `3PSL`, m_{true} is the number of true mappings out of those output and m_{tot} is the total number of mappings between G_{aux} and G_{san}. Accuracy and coverage are important metrics to evaluate the quality of mappings and a balance is needed to

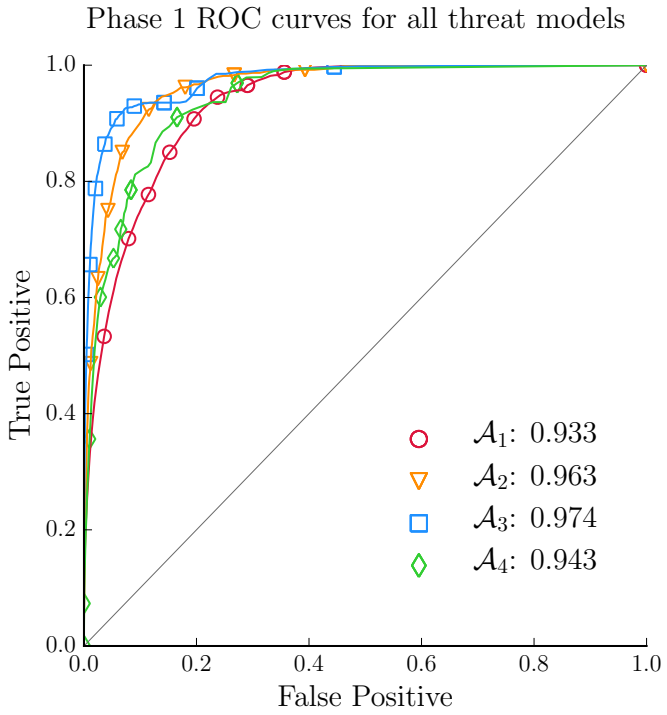


Figure 5.2: The ROC curve (AUC in legend)

Table 5.5: Phase 1: *False Positive* vs. *True Positive*

False Positive	0.001%	0.01%	0.1%	1%	10%	25%
Flickr ($\alpha_E = \alpha_V = 0.33$)	2.03	5.55	13.17	32.53	74.90	95.12
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	1.58	5.14	14.36	46.38	90.25	98.16
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	4.96	13.59	29.96	64.58	93.25	98.51
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	0.41	2.33	11.35	35.57	81.87	93.64

Table 5.6: Number of mappings produced

	Phase 1	Phase 2	Phase 3	Final
Flickr ($\alpha_E = \alpha_V = 0.33$)	8534	5668	2743	16 945
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	721	1189	1774	3684
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	997	2791	1734	5522
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	1407	2077	7555	11 039

produce good performance. Tables 5.7 and 5.8 show the accuracy and coverage of the mappings produced with a comparison presented in Figure 5.3. Results show that our algorithm performs well and produces high overall coverage and accuracy. The percentages for Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$) are lower because of the graph being very sparse, as well as low edge and node overlap among auxiliary and sanitized graphs making it hard

to attack. Figure 5.4 shows the behavior of accuracy and coverage with increasing node degree; as more information is available due to increase in degree the node pairs become easier to attack. We also study the behavior of accuracy and coverage of the output mappings by varying classifier threshold. Figure 5.5 illustrates that while the accuracy remains roughly constant the coverage drops with the increase in threshold. A drop in coverage is natural as a lower fraction of node pairs have a high classification score.

Table 5.7: Mapping accuracy (%)

	Phase 1	Phase 2	Phase 3	Final
Flickr ($\alpha_E = \alpha_V = 0.33$)	95.85	92.36	66.10	89.87
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	74.06	37.85	6.82	29.99
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	96.69	85.78	55.36	78.20
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	86.99	71.74	25.31	41.91

Table 5.8: Mapping coverage for node degree above t_3 (%)

	Phase 1	Phase 2	Phase 3	Final
Flickr ($\alpha_E = \alpha_V = 0.33$)	94.20	79.22	65.24	84.27
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	70.36	36.47	15.45	39.81
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	95.63	87.79	59.89	80.89
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	83.72	67.30	22.88	38.44

5.4.2 Evolution of mappings

The mappings evolve as we iterate using the new mappings every time to filter node pairs based on cosine similarity of common friends starting with the initial mappings. Figure 5.6 shows that accuracy and coverage are always low at the start but after enough iterations they stabilize. Figure 5.7 depicts similar behavior for the total number of mappings. Discovery of mappings in Phase 1 ensures quick convergence in Phase 2 and Phase 3 with mappings stabilizing faster. The nature of the graph also affects the convergence; Flickr ($\alpha_E = \alpha_V = 0.33$) converges faster as group membership helps re-identify node pairs whereas Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$) being sparser with low overlap takes the longest to converge. Increasing the Epinions overlap improves the adversary's side information and makes it stronger and more successful as seen by quicker convergence. We perform an additional step for Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$) due to the low number of mappings; after completing Phase 2 we reuse the Phase 2 mappings along with Phase 1 mappings to run Phase 1 again, this improves the accuracy and coverage slightly after which we discard the old Phase 2 mappings and proceed as usual, only the first run is reported here.

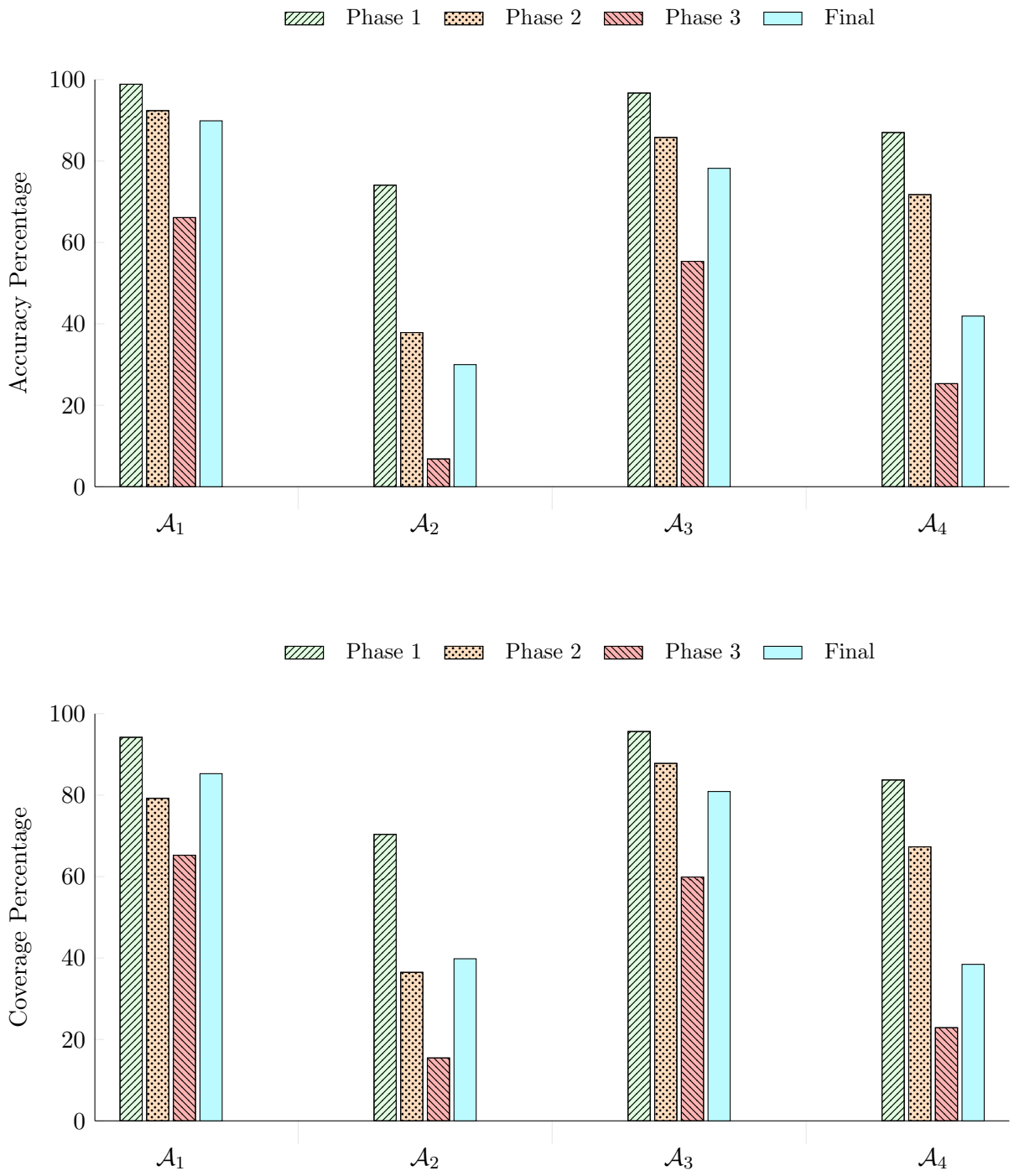


Figure 5.3: Success of adversary

5.4.3 Error analysis

In this section we analyze the errors that our algorithm makes. Phase 3 node pairs are the hardest to re-identify due to being low degree and consequently having insufficient information. Figures 5.8 to 5.10 illustrate a heat map of the joint degree distribution of true, false and unidentified node mappings. The figures show that most false mappings are concentrated among low-degree node pairs while the true mappings are spread out

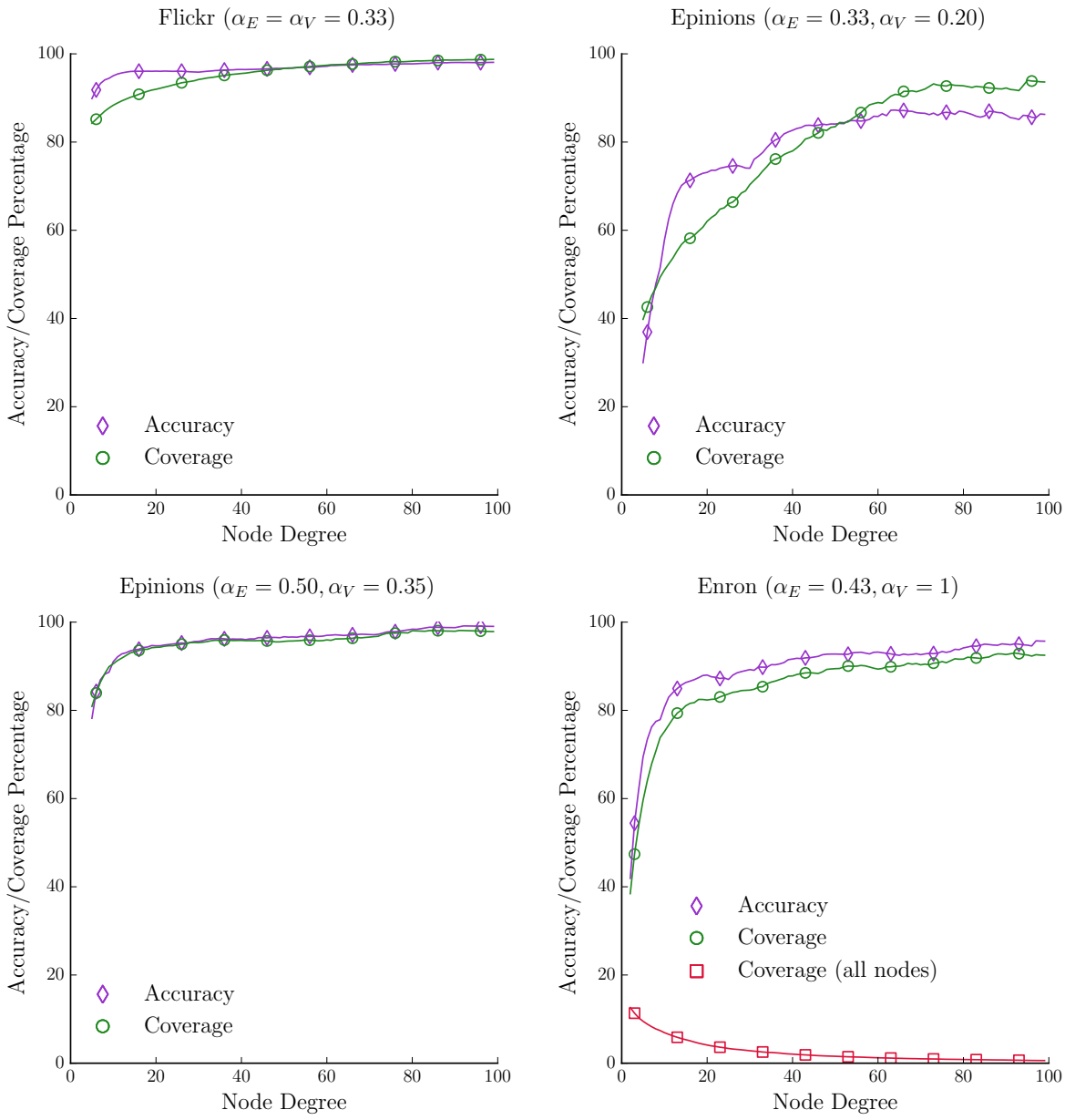


Figure 5.4: Variance of accuracy and coverage with degree

across the diagonal. Mappings that could not be identified are also concentrated among the low-degree nodes with a light spread along the diagonal but their overall numbers are low compared to other mappings. A large number of true mappings are also concentrated among the low-degree nodes since such node pairs are large in number and overwhelm the mappings in high node degree vicinity.

We study the proximity of generated mappings by merging G_{aux} and G_{san} using the common nodes to create the complete graph G_{comp}. We measure the shortest path length between the generated mappings in G_{comp}. A true mapping produces a shortest path length of 0 due to the source and the target being identical. For each phase we plot the various path lengths as a percentage of the total mappings produced. Figures 5.11 to 5.14 reveal that the output mappings are always in close proximity, with almost all mappings

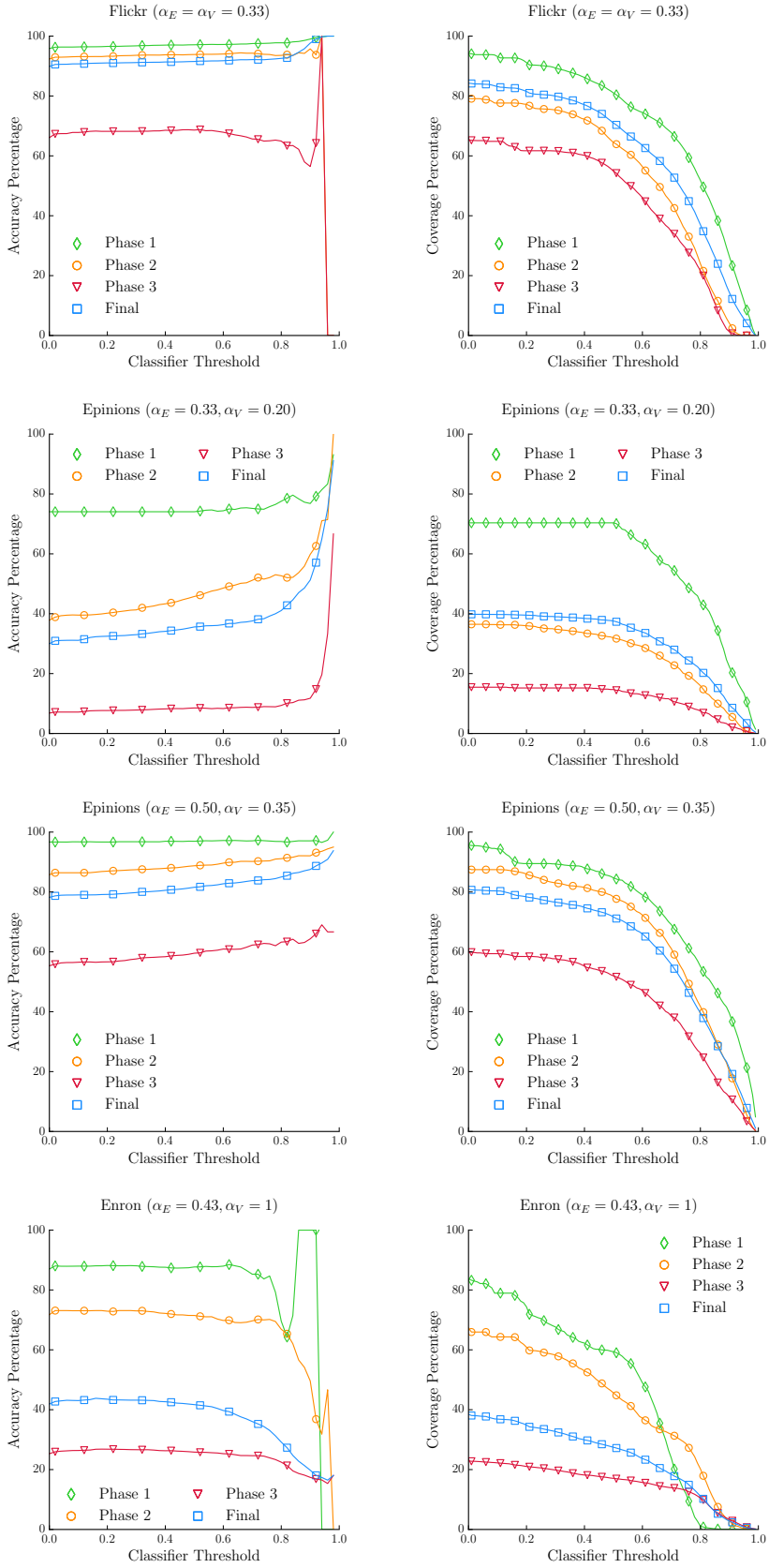


Figure 5.5: Variance of accuracy and coverage with classification threshold

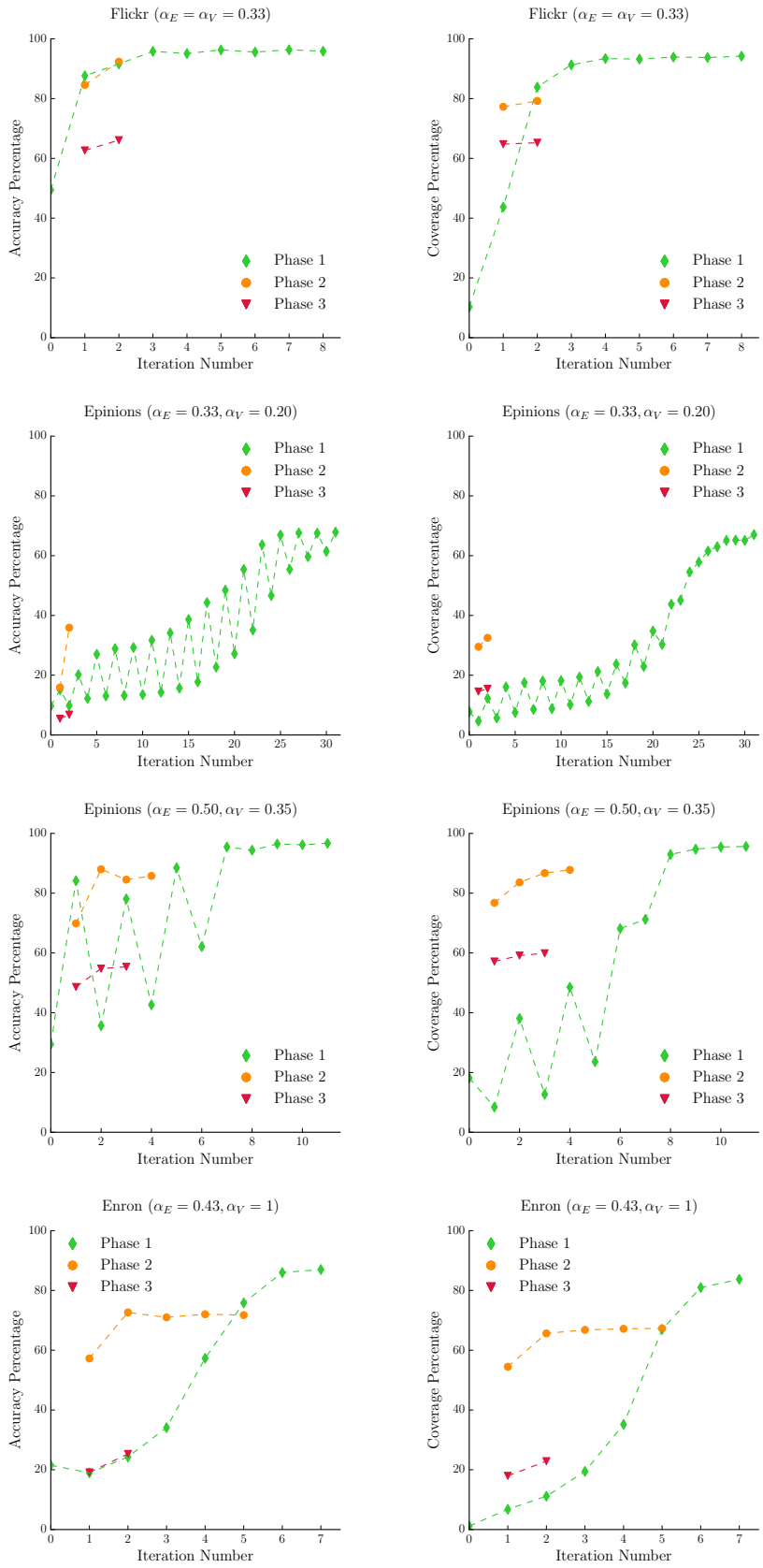


Figure 5.6: Variance of accuracy and coverage on iterating

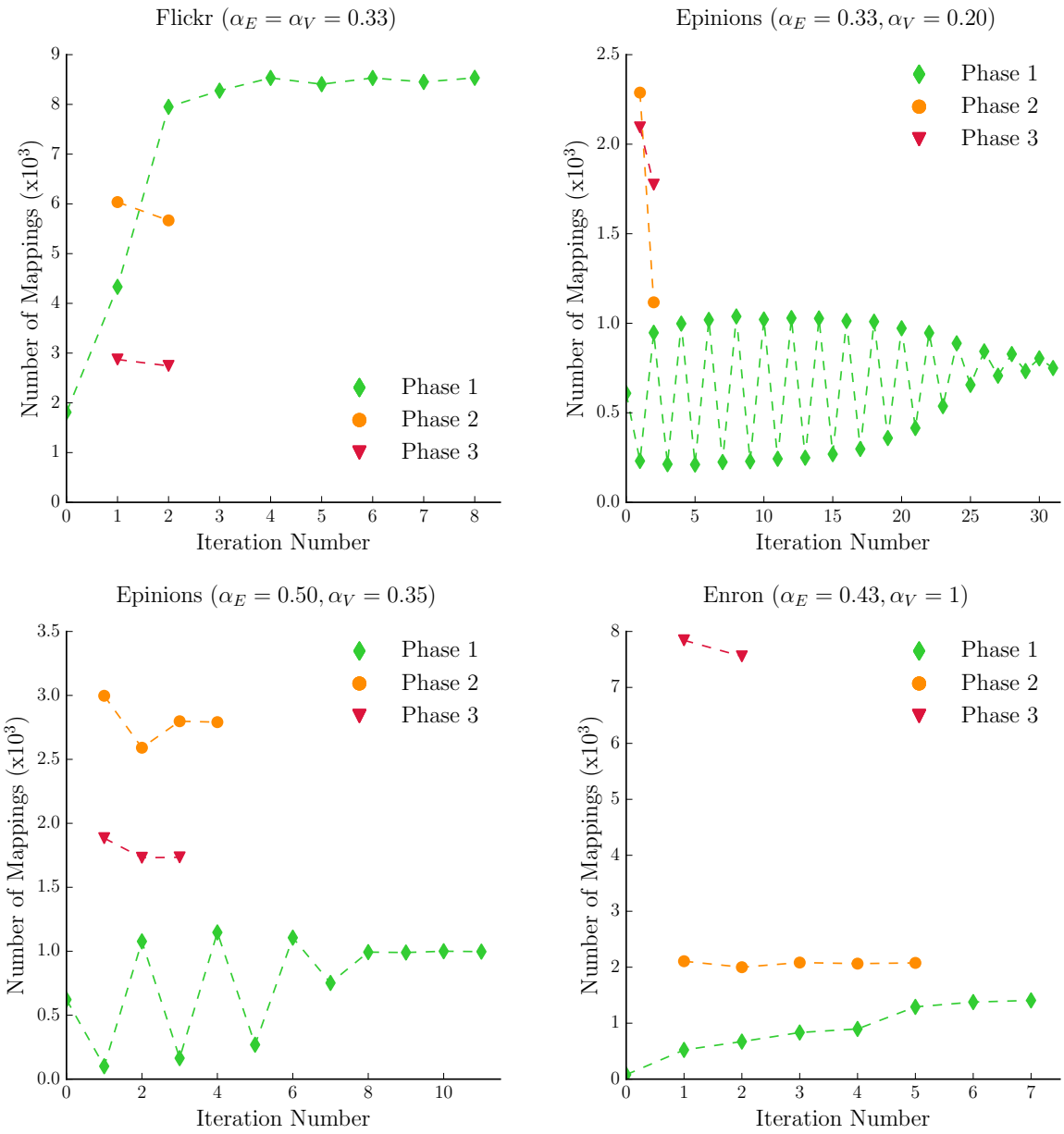


Figure 5.7: Variance of mapping count on iterating

being within two hops of each other. Since the node features are extracted from 2-hop neighborhoods it is not surprising that almost all node mappings lie within this radius. It is highly unlikely for distant nodes to be matched and this is confirmed by the results. Our algorithm is most successful for Phase 1 node pairs and least successful for Phase 3 node pairs. The neighborhoods of low degree nodes are less diverse as compared to high degree nodes, moreover, the number of low degree node pairs at a distance of two hops from each other is a lot higher than node pairs at a distance of one hop. Hence, when the algorithm makes an error it is more likely to do so for low degree nodes at a distance of two hops from each other. This explains the reason for a larger percentage of errors in the region of two hop distance; the error percentage increases with graph sparsity as we lose diversity as observed in the difference between performance of $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ and $\mathcal{A}_4$. Sparse

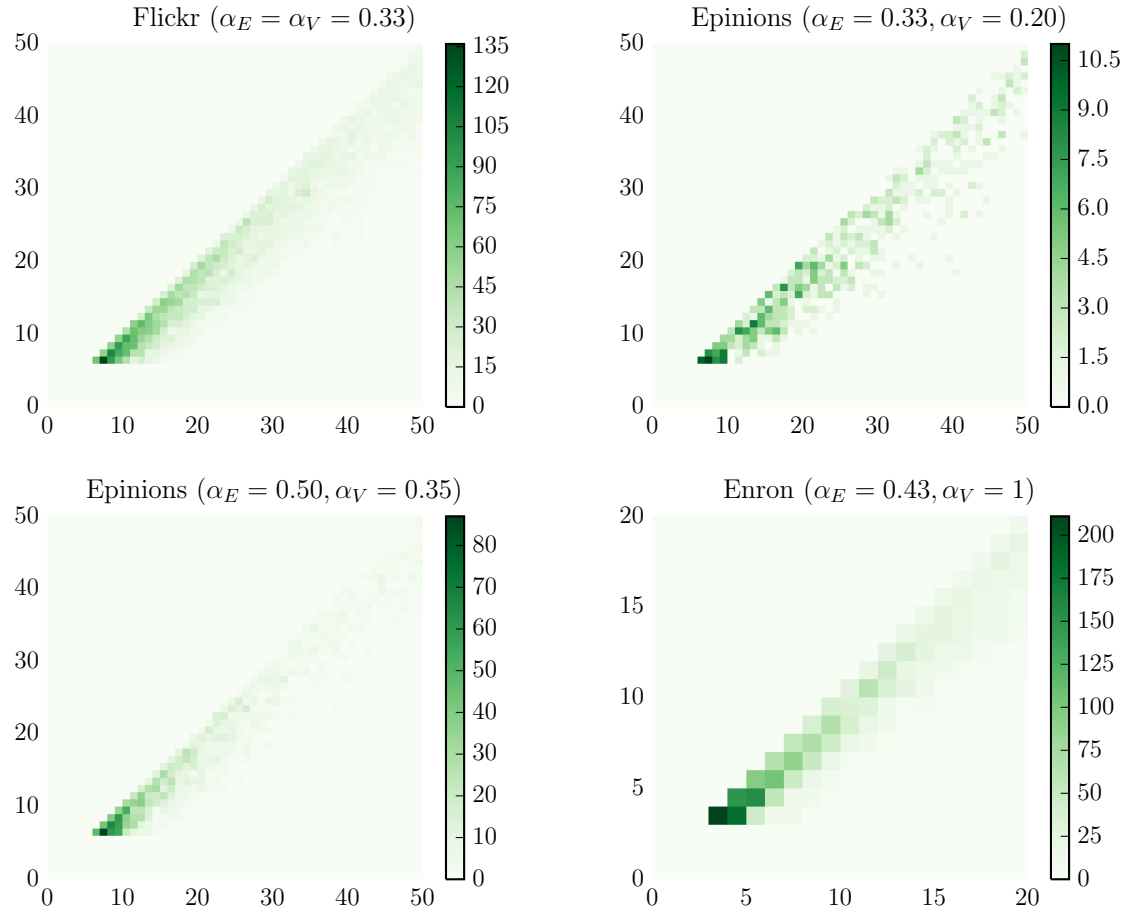


Figure 5.8: Joint Degree Distribution: True mappings

graphs also have fewer Phase 1 and Phase 2 node mappings thus confounding filtering of low degree node pairs.

Node pairs with a higher neighborhood overlap are more likely to be identified. Figure 5.15 shows the cumulative percentage of node pairs above a given neighborhood overlap is considerably higher for node pairs that are correctly identified. The neighborhood overlap is measured as the Jaccard Coefficient (see, Equation (2.1)) of the 2-hop neighborhoods of the nodes of a pair. The mappings that are falsely identified as well as those that could not be identified share a very similar low neighborhood overlap which induces errors from the algorithm. An effective way to defeat structural attacks would be delete enough edges until the overlap becomes low – the downside being that such graphs would then probably be unfit for any meaningful study.

5.5 Comparison with the state of the art

In this section we measure how well our attack fares as compared to other prominent ones. Ji *et al.* [102] conduct a survey of graph anonymization and de-anonymization strategies in

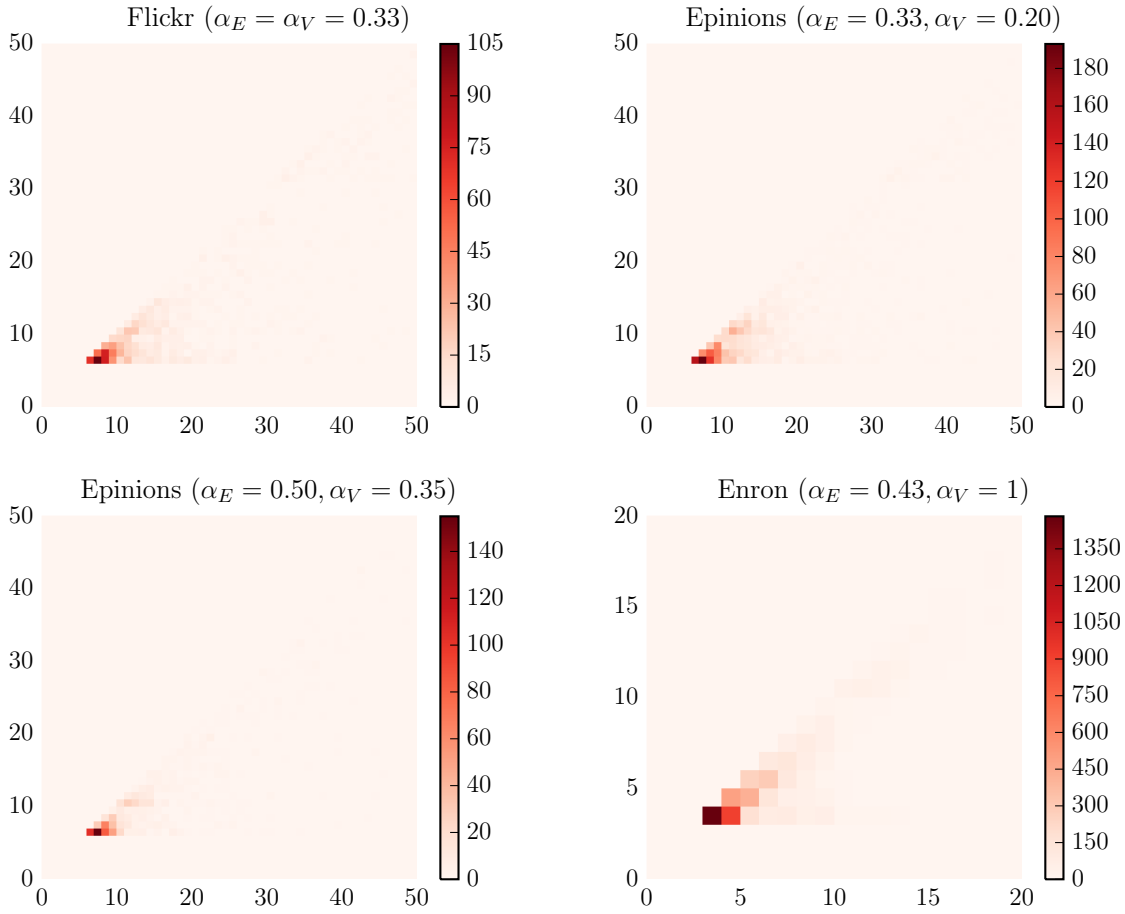


Figure 5.9: Joint Degree Distribution: False mappings

which they evaluate the de-anonymization attacks using the Enron and Facebook datasets under $\mathcal{A}_4$'s adversarial model; we compare `3PSL` with the reported evaluation. We consider all the passive seed-based and seedless attacks for comparison, active attacks are excluded as they do not scale well and require considerable control on the part of the adversary. We exclude `NKA` as it is not an attack on its own and meant to enhance existing attacks. We also exclude the passive attack of `BDK` as it does not work on perturbed graphs. All seed-based attacks are provided with 50 seed mappings [102]. The distance-vector based attack of `SH` is used for comparison as all other attacks do not scale even with pre-identified seed mappings. Our comparison uses the graph-overlap estimating attack of `JLS+`. Ji *et al.* [102] set $m_{out} = m_{tot}$ in their experiments[4], hence accuracy is equal to coverage (see, § 5.4.1).

We compare the performance of attacks based on Enron and Facebook (nodes = 63 731, edges = 817 090) datasets same as those used by Ji *et al.* [102]. A shoulder to shoulder comparison is presented by measuring the coverage based on all node mappings, not just those attacked. We do not attack nodes of degree below three for Enron, doing so would increase the coverage but it would come at the cost of accuracy. Figure 5.4 shows the

[4]Confirmed via communication with the authors.

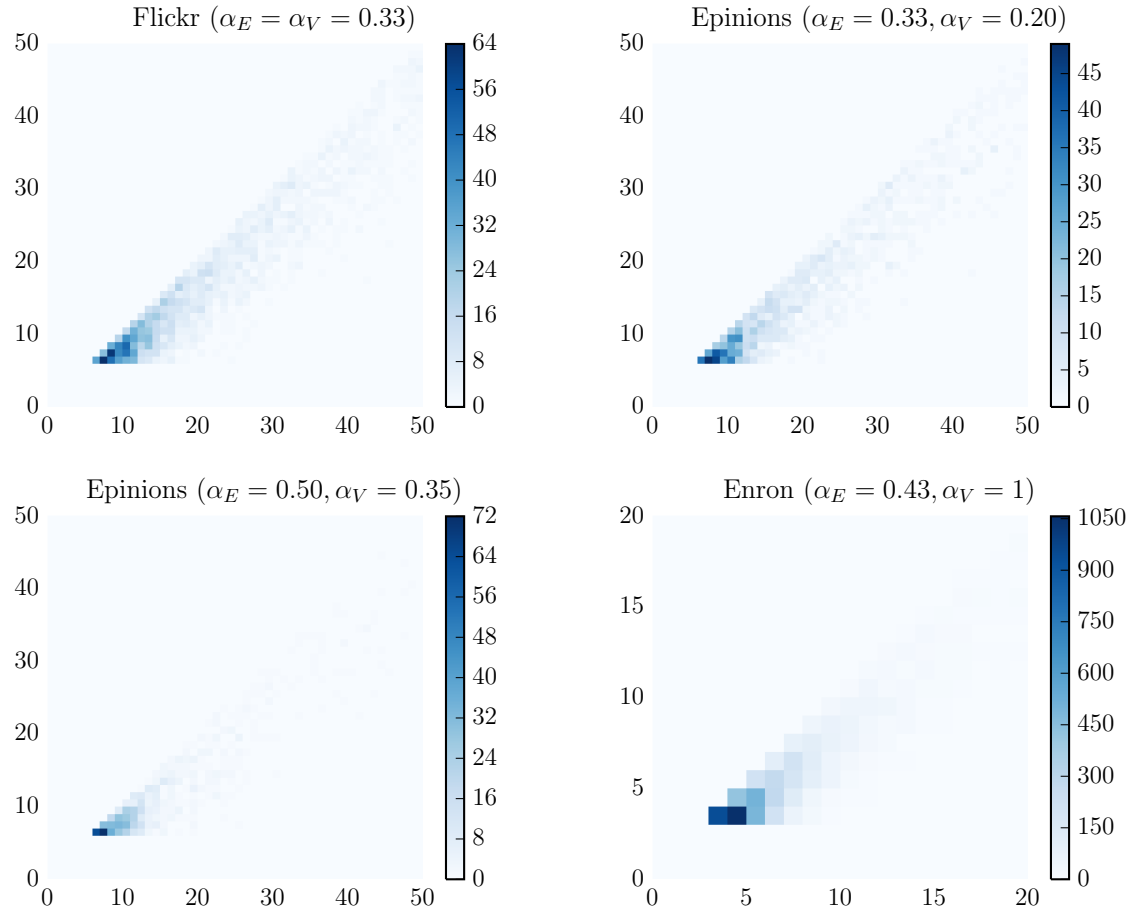


Figure 5.10: Joint Degree Distribution: Unidentified mappings

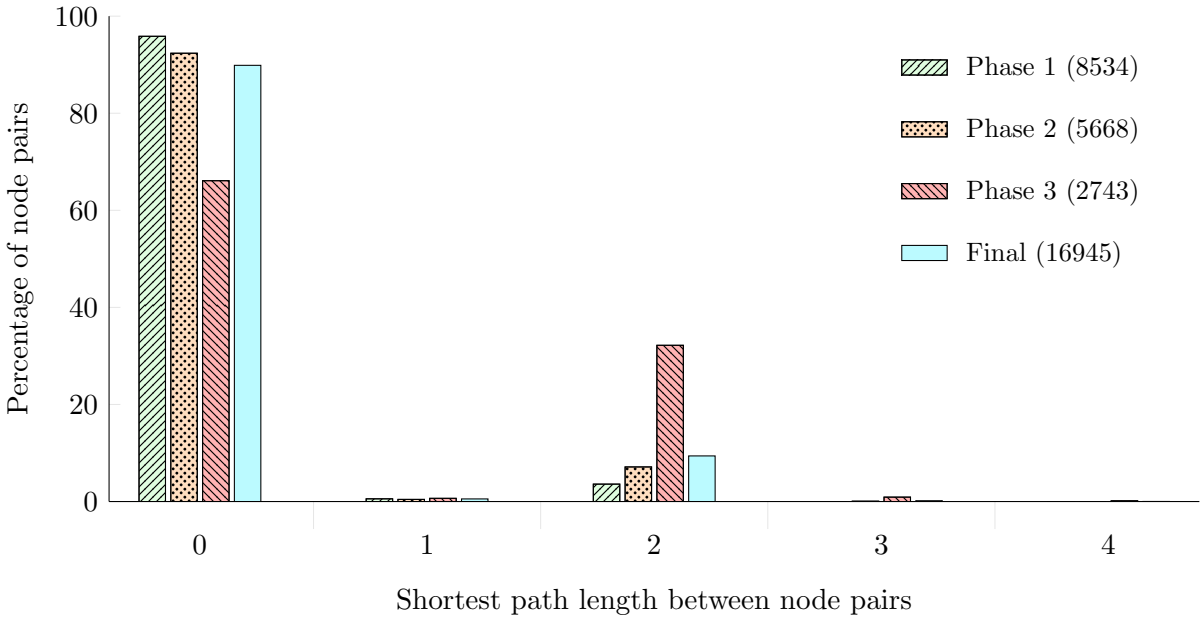


Figure 5.11: Flickr ($\alpha_E = \alpha_V = 0.33$)

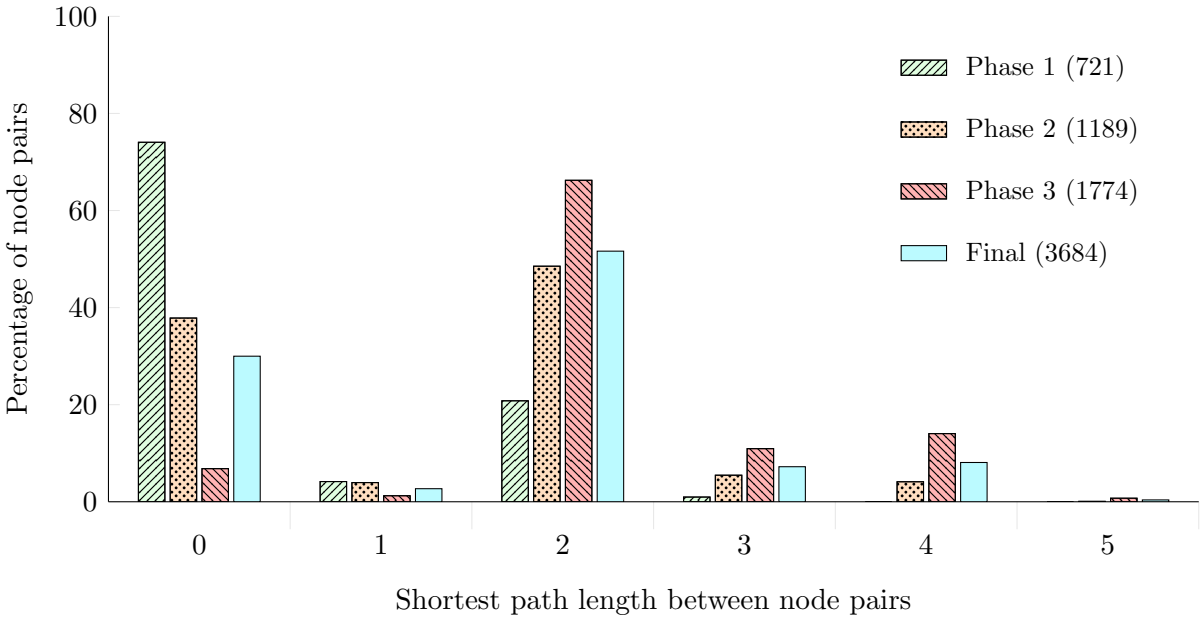


Figure 5.12: Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)

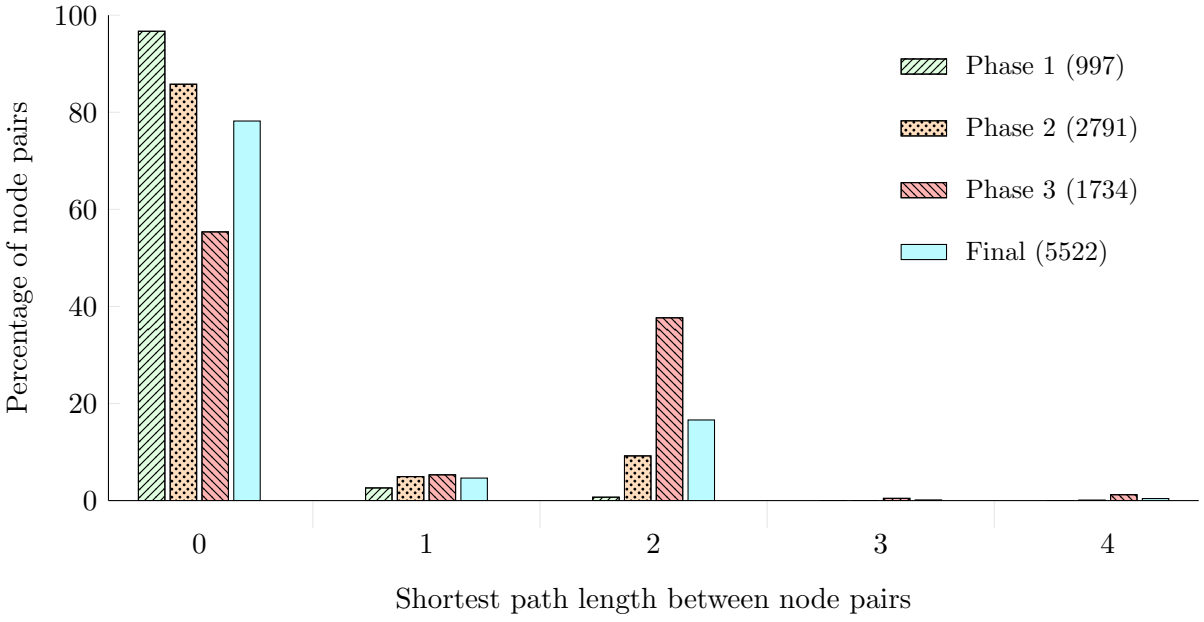


Figure 5.13: Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)

degradation of Enron's coverage computed for all nodes as degree is increased. Facebook is more dense than Enron and has a higher average node degree; as seen from § 5.4 and the attacks presented in §§ 2.8.1 and 2.8.2, this increases adversary's success [102, 104]. We estimate a lower bound for the overall accuracy and coverage for Facebook by using the values for Enron.

Table 5.9 presents a thorough comparison of all the attacks. We see that despite attacking only nodes of degree greater than two and no seed knowledge `3PSL`'s overall performance is better in all scenarios by a large margin. `KL` is the best attack for Enron, even with seeds

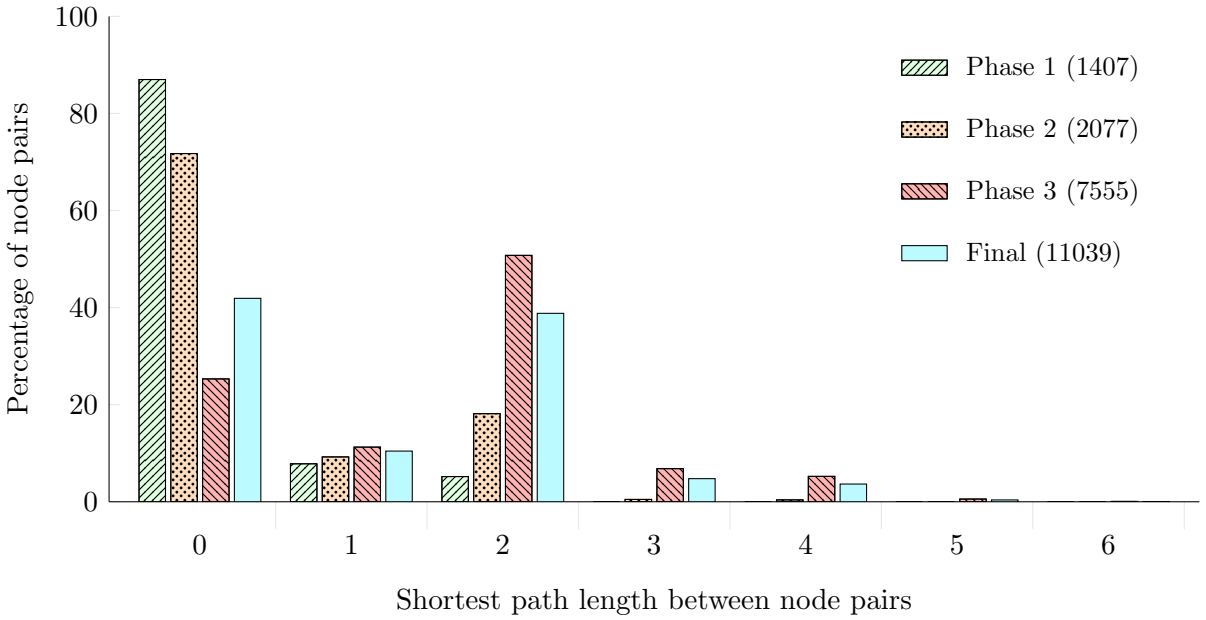


Figure 5.14: Enron ($\alpha_E = 0.43, \alpha_V = 1$)

it only achieves a coverage and accuracy of 15.96% marginally better than our coverage of 12.61%, we achieve an accuracy of 41.91% which is over 2.5 times than that of KL. The best seedless attack on Enron fares even worse, with coverage and accuracy of 11.91%; comparatively our algorithm has higher coverage with almost four times the accuracy. The best attack for Facebook is YG which uses seeds to achieve a coverage and accuracy of 28.32% our algorithm achieves a coverage of over 40% and more than twice the accuracy of over 65%. The best seedless attack for Facebook achieves a coverage and accuracy of 14.73%; comparatively our algorithm achieves more than twice the coverage with over four times the accuracy. Seed knowledge is critical to attack performance in sparse graphs (Enron) vis-a-vis dense graphs (Facebook). As adversary's access to structural information is limited, the dependence on seeds increases. Thus gap between seedless and seed-based attacks is larger for sparse graphs like Enron as compared to denser graphs like Facebook. Even in such an adverse scenario our algorithm shows better overall performance for Enron with only slightly lower coverage (compared to seed-based attacks) to achieve a drastic gain in accuracy, that too without seed knowledge. For Facebook, which is very dense, no other attack comes even close to the performance of our algorithm even with seed knowledge. The attack on Facebook is more successful as a larger fraction of node pairs fall under Phase 1 and 2 as compared to Enron which are easier to attack. In particular, only 32.79% of Enron and 65.18% of Facebook nodes have degree over two and attacked by our algorithm. We can increase the coverage of our algorithm with a decrease in accuracy by attacking lower degree nodes. It is better to have a low coverage and high accuracy thus producing some useful mappings rather than high coverage and low accuracy which would be akin to random guessing. 3PSL's accuracy is appreciably higher than all other attacks including those that use seed knowledge.

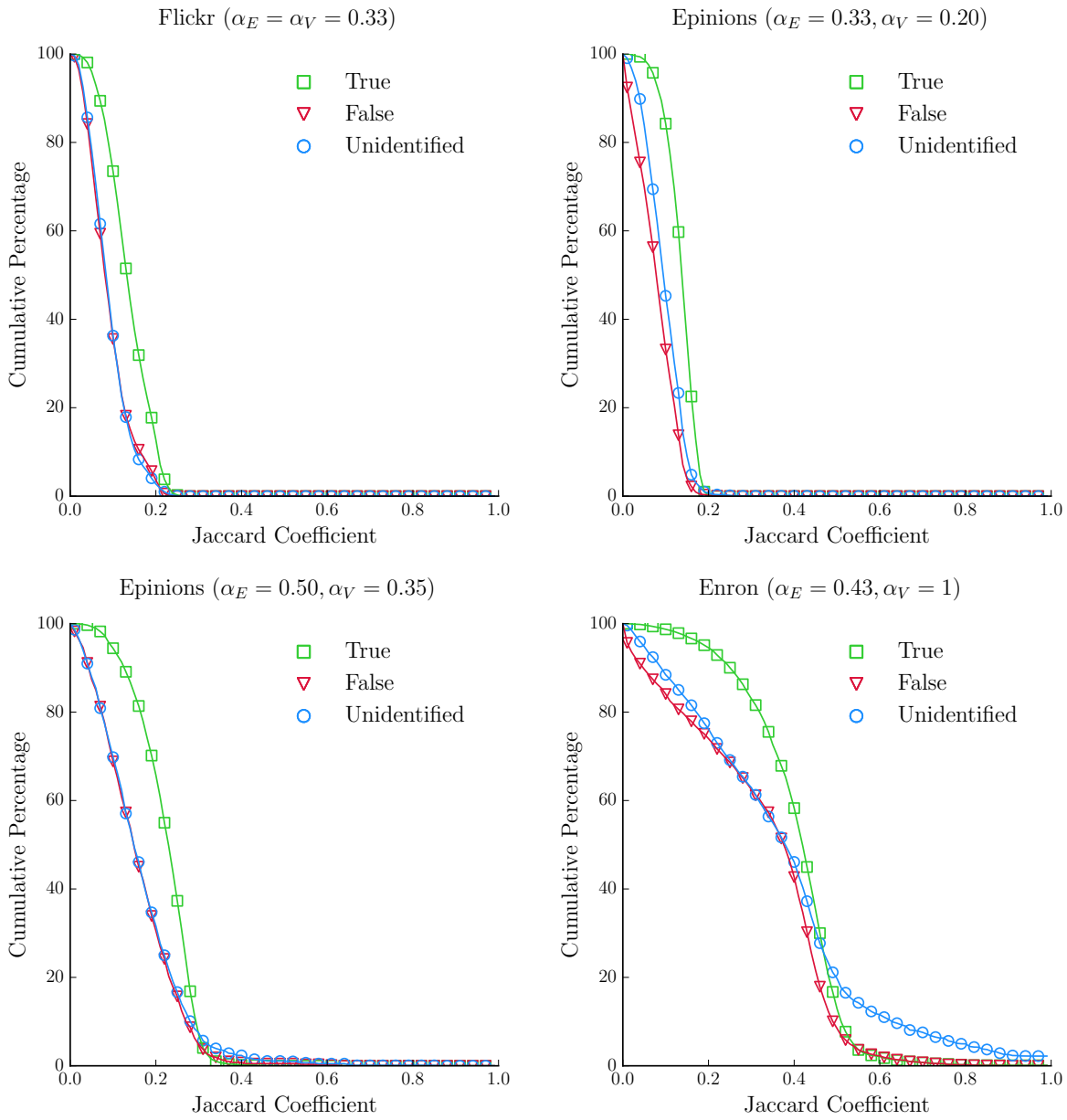


Figure 5.15: Cumulative percentage vs Jaccard Coefficient

5.6 Discussion

As demonstrated by §§ 5.4 and 5.5, `3PSL` shows a marked improvement over all other graph de-anonymization algorithms. Not only does it perform better but it can also accommodate a variety of adversaries and does so while using less information than all the seed-based attacks. The attack is agnostic to the change of anonymization scheme and trains a model using data samples generated from anonymized graphs. The success of our attack emanates from the fact that we not only consider structural similarity but also similarity based on common friends to recover mappings. These metrics are orthogonal and thus complement each other perfectly. Our approach stands in contrast to schemes relying only on similarity metrics based on common neighbors [114, 115]; it achieves improved

Table 5.9: Comparison of coverage and accuracy percentage with other attacks (accuracy = coverage for all attacks except 3PSL); $^{\varnothing}$ denotes a seedless attack. A higher percentage is better.

	Enron		Facebook	
	Coverage	Accuracy	Coverage	Accuracy
KL	15.96	15.96	5.99	5.99
JLS+	13.05	13.05	15.68	15.68
SH	12.77	12.77	15.63	15.63
JLSB$^{\varnothing}$	11.91	11.91	14.73	14.73
YG	3.10	3.10	28.32	28.32
PFG$^{\varnothing}$	7.39	7.39	10.87	10.87
NS	0.37	0.37	0.18	0.18
3PSL$^{\varnothing}$	12.61	41.91	>40	>65

performance while attacking graphs with low node overlap.

3PSL outperforms the previous generation of algorithms, all of which are based on hand-curated heuristics and thus prone to human bias and poor judgment. The high coverage and accuracy for Enron (coverage = 12.61%, accuracy = 41.91%) and Facebook (coverage > 40%, accuracy > 65%) is achieved by training statistical models to take decisions in the presence of huge amounts of data which is overwhelming for the humans. In the future these attacks could be further improved by using better feature vectors and learning models, hence these results should be considered as a lower bound when estimating likelihood of re-identification. In our work we focus on developing a machine-learning approach to de-anonymize graphs rather than finding the optimum tuning parameters which is quite a challenging task in itself. As techniques are refined the attacks will get stronger as they always do.

The results demonstrate that a typical social network like Facebook is quite vulnerable to re-identification attacks with close to half of the individuals with more than two friends being identified with over 65% accuracy. Most people in a social network have more than 25 friends and for them both coverage and accuracy would be appreciably higher in the given adversarial model. The threat is even stronger when we consider that we restrict the adversary to only have access to the graph topology; such attacks can be considerably improved if the adversary is a member of the social network or could collect side information about the target. This calls for exercising great caution when releasing such datasets publicly. Automating attacks strongly influences the economics of security by removing human input to a large extent. This allows the attacker to run large scale attacks without much intervention thus decreasing the cost significantly. Even when

the output mappings are wrong the errors are in the vicinity of the target node which may be enough to breach privacy depending upon the attack scenario. In summary, our experiments indicate that a data subject whose social graph is released cannot be provided with reasonable privacy guarantees. If the adversary possesses even imperfect knowledge of the target's neighborhood then re-identification should be considered an imminent threat. Hence, it would be prudent on the part of data providers to be transparent about the privacy expectations an individual should have when their data is released.

Seed quality. All seed-based attacks assume knowledge of seeds to prime the de-anonymization process. This may be considered reasonable in certain scenarios, but possession of error-free seed mappings is a very strong requirement. As seen in § 2.8.1, algorithms impose additional requirements on the seed set such as their size, structure, degree, centrality and neighborhood. These requirements along with the size and structure of the graph under attack heavily influence the quality of the seeds and as a result do not generalize well [173]. Such graph de-anonymization algorithms are finicky as their success is very sensitive to the size and quality of the seed set, and below a critical size the attack fails to converge [115, 174, 175]. Table 5.9 shows that the performance of all seed-based attacks (`KL`, `NS`) does not improve when attacking Facebook as compared to Enron; seed selection has a huge impact on the results. Seed-based attacks are even less suitable when the node and edge set overlap of auxiliary and sanitized graphs is low. Exploiting seeds could lead to powerful de-anonymization attacks, though reliance on seeds is a serious limitation. Ji *et al.* [116] demonstrate that structural attacks can be more potent than seed-based attacks, which is corroborated by our results thus tilting the scales further against relying on seeds.

Parameter choice. Classifying node pairs is more expensive than filtering them, hence the choice of t_1 influences the overall efficiency of the algorithm. It is also important to partition node sets in such a way that trained decision forests can be focused to maximize accuracy due to similar properties of the data points. Choosing t_1 too low makes the data points too varied, whereas setting it too high yields too few data points to effectively train the model; for this reason sparser graphs have a lower t_1. Flickr has a lower t_1 despite being dense because we could select node pairs based on identical group membership, this allows us to get more high quality mappings for Phase 1. Identifying node mappings in a phase helps reduce number of node pairs to be tested in subsequent phases hence it is desirable to identify a large number of nodes in Phase 1, t_2 and t_3 are identical for all datasets except for Enron we chose a lower t_3 to increase coverage and enable comparison. We set vector length and bin size (n, b) to be $(7, 50)$ and $(30, 35)$ for directed and undirected graphs respectively (see, § 5.3.2). The value of (n, b) is chosen such that it can accommodate higher degrees and their variation, the choice does not have a huge impact on accuracy (see, § 3.5.3). A low cosine similarity threshold is chosen to filter out most spurious node

pairs so that the remaining ones can be cleaned using the classifier; non-identical node pairs with sufficiently high degree tend to have very low cosine similarity. As discussed in § 3.5.3 we use forest size of 400 for good testing accuracy. Phase 1 mappings stabilize after about 10 iterations which can be increased further to ensure stability as each iteration is cheap at this stage. Subsequent phases do not require many iterations due to sufficient mappings being found in Phase 1.

Feature selection. We experimented by using features such as centrality, edge weights and group membership in addition to those proposed. Complicated features do not provide significant improvement. Perfect knowledge of groups does not improve the ROC curve by much either; this is due to the fact that identical group membership is already captured to a large extent in the 2-hop neighborhood degree distribution. The technique of using groups to improve attacks (`NKA`) is useful for schemes that use only local features but it does not help much in our case. Moreover, the knowledge of group membership is not likely to be precise in reality which further dampens their effect. Even if using complicated features were to provide a significant improvement, it would not matter a lot in the grand scheme of things. We rely on the classifier's success at selecting node pairs that are more likely to be true; beyond a certain point the classification quality does not make a major difference. It is more important to improve the filtering process (see, § 5.3.3) as it makes the algorithm more efficient by improving the ratio of true vs false node mappings and decreasing the number of mappings to be classified. We also experimented with features beyond two hops but this increased false positives, as using large subgraphs to represent a node produces overlaps with many other nodes.

5.7 Summary

This chapter showed the difficulties in manually optimizing parameters to find the optimum graph de-anonymization attack and how automated learning models can aid these choices. We demonstrated an attack based on random forests that outperforms all other attacks that too while using simple and efficient node features. The attack proceeds in phases and uses complementary node similarity metrics derived from common friends and structural similarity to identify nodes across graphs. The high level of coverage and accuracy achieved by our attack even for low degree nodes suggests exercising restraint when publishing social network datasets. Automation of attacks has a strong effect on the security economics as it significantly lowers the adversary's cost to breach privacy at scale. Even partial knowledge of a target's neighborhood may be sufficient to compromise privacy specially when it can be combined with readily available side information. We believe our work can usher in a new era in social graph de-anonymization.

Chapter 6

Conclusions

In this dissertation we have proposed a fresh approach to social graph de-anonymization by casting it as a learning task. This opens up the possibility of analyzing the problem from a new angle and utilizing machine learning tools to solve it. Our work focused on the following three areas.

Chapter 3 laid the foundation for the learning model used throughout this work. The D4D challenge in the Ivory Coast released call detail records of individuals and provided the initial motivation for our work due to the privacy questions it raised. Privacy leaks could have serious consequences when they pertain to countries like the Ivory Coast which has a history of civil wars and political unrest. Thus carefully evaluating the privacy of data anonymization schemes, such as the ones made available by the D4D competition is imperative. Traditionally such an analysis had to be performed manually, and painstakingly repeated for any new variant of the anonymization scheme, despite general results indicating that social network anonymization schemes are likely to be broken.

Our approach cuts the need for manual effort, and can uncover artefacts of the anonymization process using examples that allow re-linking nodes in "anonymized" networks. Unraveling an anonymization strategy used to be a challenging task that took non trivial manual effort and time. However, devising a new strategy was not as demanding, and even poorly designed anonymization methods could take a significant amount of work to break. For example, we presented an attack on Scheme 1 that works well for 1-hop node pairs and gave mixed results for others. However, the trivial deletion of 2-hop links in Scheme 2 disrupts the invariant relied upon for this attack, without necessarily guaranteeing security. In terms of the specific anonymization procedures for the D4D competition that motivated the study: we do conclude that Scheme 2 is marginally harder to de-anonymize and re-link than the original Scheme 1 we evaluated. Yet neither provides a level of privacy we would recommend for public release of data. Even a reliable linkage rate of 1% would lead to a significant fraction of the mobile operator's customers being potentially linked. Big data make even a relatively low probability of linkage events certain to harm someone. In fact

for a number of realistic configurations we show the linkage rate was much higher, leading to efficient attacks particularly if some side information is available to build good priors and tolerate slightly higher false positives. Thus we are relieved that the competition only released such data under confidentiality agreements.

Our approach, and the learning task we rely upon, to link or de-anonymize, are purposefully simple: they only use graph topology, not attributes, directionality or multiple snapshots of graphs over time, despite the availability of such data in some cases. They also consider a single pass of the procedure, where no known seeds are available. It is clear that richer features could be used, and the attack can be iterated once a handful of nodes have been uncovered to unravel larger graphs. As shown in later chapters such attacks could be mounted. Our results confirm previous wisdom that releasing anonymized social network data is likely to result either in a privacy catastrophe, or very poor utility. In fact it is clear that releasing egonets through Scheme 2 leaks a significant amount of information, but manually uncovering new invariants for it would require appreciable amounts of work. A further trivial modification would again void the new analysis. Deleting information of nodes as a function of their location vis-a-vis the ego leaks information and is flawed as an approach. Our attack uses only graph topology to re-identify individuals in varying sub-graphs and runs in linear time complexity in the number of node pairs classified. Using machine learning provides the advantage of amortizing our efforts and eliminates the need to construct attacks from scratch. Modularity ensures that the features can be picked in accordance to the anonymization strategy employed to get better leverage. Our methods are agnostic to the anonymization strategy employed and present a clearer understanding of data.

The versatility of the machine learning techniques is further demonstrated in Chapter 4, where we focus our attention on benchmarking social graph anonymization techniques. It has always been easy to propose graph anonymization schemes, but harder to assess whether they actually work. We provide a framework that levels the playing field for the first time by automating the analysis of such schemes. Quick and automated analysis empowers data holders to swiftly triage newly proposed schemes. We show how to train a classifier in the absence of ground truth by generating subgraphs and sampling data from auxiliary and sanitized graphs. The classifier uses node features that can adapt to changes in adversarial model (as demonstrated for `1HKA`) and accommodate adversaries of varying strength. Incremental features allows us to model adversaries with much more sophisticated queries based on neighborhood and can be easily modified to include node and edge attributes. Traditional structure-based graph attacks cannot readily adapt to changing adversaries. Additionally, using attacks to compare schemes is not ideal as vulnerability to a particular attack does not capture the true extent of a scheme's failure, since near misses are not considered when producing full mapping. Measuring true positive versus false positive rate is far more granular and gives us more information. Unlike attack based measurement our framework does not assume a model of the adversary which can

tolerate only a certain amount of error and hence rigid. We perform a detailed analysis of six perturbation-based social graph anonymization schemes. A thorough study of trade-off between anonymity and utility as a function of graph perturbation is also presented.

Our experiments conclusively show that none of the schemes analyzed provide acceptable levels of anonymity while retaining utility. The levels of perturbation at which some privacy guarantee is achieved destroys most utility. Schemes that introduce global perturbation in graphs are better suited to preserving utility as well as anonymity. Any attempt to preserve only a particular metric of the graph or perturbing it in a formulaic way produces poor results both in terms of anonymity as well as utility. Our adversarial model considers an adversary which has access to imperfect structural information of the graph which is used to identify members of intersecting graphs. Both the graph at the adversary's disposal and the released graph are aggressively and synthetically damaged which limits the adversary. In practical situations it is highly likely that the adversary can get hold of a graph that has been damaged by organic processes that are not adversarial. In such a scenario the attacks will be more potent and catastrophic for the privacy of individuals whose data is released.

We believe taking a conservative approach while releasing data is the best way forward. Even though anonymization schemes are useful in that they can be used to dissuade the curious but honest adversary, they must not be assumed to be able to stop a malicious adversary if utility preservation is important. Social graph anonymization schemes and anonymization schemes dealing with high-dimensional data in general should always be backed up by legal agreements which prohibit malicious use of data. There is a place for anonymization schemes in social graph research but it does not belong in the realm of preserving privacy.

Finally, Chapter 5 presents an end-to-end de-anonymization attack that builds on inferring whether pairs of nodes are identical or not, to recover node mappings across full graphs. Adversarial machine learning and de-anonymizing behavioral patterns are two sides of the same coin and they are converging fast. In the presence of big data, attacks based on heuristics will gradually be replaced by learning models because of their adaptability, automation and superior performance. Not only are automated models better but as demonstrated in Chapter 3, the learning is transferable across datasets [159], this provides a significant improvement over the traditional techniques. It is not surprising that learning models can surpass human intuition although the margin of improvement is startling. Our work shows how to approach the problem of social graph de-anonymization in a systematic manner. We presented an algorithm that outperforms all the other graph de-anonymization algorithms proposed so far while using much stringent adversarial models without seed knowledge. The algorithm is not dependent upon heuristics for its success and uses the simple classification task of categorizing node pairs as identical or non-identical across graphs. It uses modular features that can adapt to a variety of adversarial models. The

machine learning model does not require knowledge of the anonymization scheme for training. De-anonymization is simple and efficient as confirmed by evaluation based on three real-world social graph datasets under four adversarial models. A thorough comparison with seven modern de-anonymization attacks using two datasets is also presented. Our algorithm achieves a coverage of 12.61% and accuracy of 41.91% for Enron dataset, the next best algorithm is seed-based and achieves a coverage and accuracy of 15.96% which is substantially lower. The difference is even more stark for Facebook where our algorithm achieves a coverage of over 40% and an accuracy of over 65%; the next best algorithm is seed-based and can only achieve a coverage and accuracy of 28.32%. Our attack shows a marked improvement over all other attacks. Optimizing parameters by training is better in adverse scenarios as human error is costlier in limited information.

Starting from the simple task of classifying a node pair we designed and built intricate models to capture a variety of learning tasks pertaining to social networks. The techniques proposed are versatile and can be used to quantify and benchmark social graph anonymization schemes and produce actual node mappings. Our work shows that even an automated algorithm can produce *good* de-anonymization attacks, thus highlighting the graveness of publishing social network data without proper checks and controls.

Future work. In this dissertation we have presented a number of enhancements over traditional social graph de-anonymization approaches. The automated models are not only adaptable and nimble but also provide better performance. However, social graph de-anonymization remains a challenging problem and there is still scope for improvement. Machine learning based approaches fare very well when ample data is available but training quality drops if the data is insufficient. Hence, attacking sparse graphs with only a few hundred nodes nodes is much harder. Availability of side-information is also a critical factor – modeled by the overlap between auxiliary and sanitized graph as measured by the Jaccard Coefficient (see, Equation (2.1)). The attack presented in Chapter 5 starts to become potent for $\alpha_V \geq 0.2$, below this overlap the mappings produced have low accuracy and coverage. Also, it remains to be seen if low degree node identification can be improved or whether we have hit a theoretical limit. We found it hard to achieve significant success in attacking nodes of degree below six at scale. Since the node similarity metric forms the core of the attacks presented, the run-time of the classifier is linear in the number of node-pairs classified as all node-pairs must be considered for the attack. Future work may consider improving upon these shortcomings. We note that sparsity of graphs, graph overlap and attacking low degree nodes are not problems exclusive to our approach. Other state of the art algorithms also face similar challenges many of which our approach outperforms. Extracting efficient and accurate features is one of the most important aspects of the attacks presented. We found it hard to improve the performance of classifier significantly by using more complex features. It would be interesting to study if simple features with better performance can be found. These could be topics for future work, and perfecting

such attacks could be a fruitful research field for years. However, pursuing too far such a research program may yield diminishing returns. Thus, it may be more beneficial to research instead alternative ways to perform social network analysis in a privacy-friendly manner, without the need for *anonymized* graphs.

Finally, it may be worth considering alternate solutions that enable privacy-friendly data analysis. As discussed in § 2.4 differential privacy is a step in the right direction. Despite being hard to setup and resource hungry it does provide provable privacy guarantees which are worth having for sensitive datasets. If datasets are not very sensitive then one might consider using an interactive setting similar to differential privacy but without any guarantees; releasing results of computations on data instead of the dataset itself reduces the scope of attacks. Solutions using cryptography have also been proposed, such solutions aim to prevent cleartext data exchange and distribute pieces of the database among various parties which participate in a multi-party computation based protocol to analyze the data. This limits the risk of full exposure of data to a single party. It is a challenge to setup such schemes and even after that they are limited in scope. However, such schemes provide rigorous privacy protection and with time they will continue to be refined with wider scope. Another measure that can be taken to decrease the chances of a privacy breach is to share anonymized data selectively while backing it up with a stringent non-disclosure agreement. We believe that handling data release wisely and managing privacy risk is the best way forward. It is clear that analysis of datasets is valuable to the society. Hence, a measured approach must be taken which balances risk to privacy of the individuals and the benefits extracted.

Bibliography

[1] R. A. Hill and R. I. M. Dunbar. Social network size in humans. *Human Nature*, 14(1):53–72, 2003.

[2] Orange. D4D Challenge. `http://www.d4d.orange.com`, 2012.

[3] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, pages 111–125, 2008.

[4] Arvind Narayanan, Elaine Shi, and Benjamin I. P. Rubinstein. Link prediction by de-anonymization: How we won the kaggle social network challenge. In *The 2011 International Joint Conference on Neural Networks, IJCNN 2011, San Jose, California, USA, July 31 - August 5, 2011*, pages 1825–1834, 2011.

[5] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, SP '09, pages 173–187, Washington, DC, USA, 2009. IEEE Computer Society.

[6] Charu C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pages 901–909. VLDB Endowment, 2005.

[7] Khaled El Emam, Elizabeth Jonker, Luk Arbuckle, and Bradley Malin. A systematic review of re-identification attacks on health data. *PloS one*, 6(12):e28071, 2011.

[8] Johann Bacher, Ruth Brand, and Stefan Bender. Re-identifying register data by survey data using cluster analysis: An empirical study. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):589–608, 2002.

[9] Anupam Datta, Divya Sharma, and Arunesh Sinha. Provable de-anonymization of large datasets with sparse dimensions. In *Principles of Security and Trust - First International Conference, POST 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012, Proceedings*, pages 229–248, 2012.

[10] Fida Kamal Dankar, Khaled El Emam, Angelica Neisa, and Tyson Roffey. Estimating the re-identification risk of clinical data sets. *BMC Med. Inf. & Decision Making*, 12:66, 2012.

[11] Xiaoxun Sun, Hua Wang, and Yanchun Zhang. On the identity anonymization of high-dimensional rating data. *Concurrency and Computation: Practice and Experience*, 24(10):1108–1122, 2012.

[12] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. De-anonymization attack on geolocated data. *J. Comput. Syst. Sci.*, 80(8):1597–1614, 2014.

[13] Tomotaka Okuno, Masatsugu Ichino, Tetsuji Kuboyama, and Hiroshi Yoshiura. Content-based de-anonymisation of tweets. In *Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP 2011, Dalian, China, October 14-16, 2011*, pages 53–56, 2011.

[14] Jayakrishnan Unnikrishnan and Farid Movahedi Naini. De-anonymizing private data by matching statistics. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing, Allerton Park & Retreat Center, Monticello, IL, USA, October 2-4, 2013*, pages 1616–1623, 2013.

[15] Joseph A. Calandrino, Ann Kilzer, Arvind Narayanan, Edward W. Felten, and Vitaly Shmatikov. "you might also like: " privacy risks of collaborative filtering. In *32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA*, pages 231–246, 2011.

[16] George Danezis and Carmela Troncoso. Vida: How to use bayesian inference to de-anonymize persistent communications. In *Privacy Enhancing Technologies, 9th International Symposium, PETS 2009, Seattle, WA, USA, August 5-7, 2009. Proceedings*, pages 56–72, 2009.

[17] George Danezis and Carmela Troncoso. You cannot hide for long: de-anonymization of real-world dynamic behaviour. In *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013*, pages 49–60, 2013.

[18] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. Trawling for tor hidden services: Detection, measurement, deanonymization. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 80–94, 2013.

[19] Bradley Malin and Latanya Sweeney. How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems. *Journal of Biomedical Informatics*, 37(3):179–192, 2004.

[20] Nicholas D. Lane, Junyuan Xie, Thomas Moscibroda, and Feng Zhao. On the feasibility of user de-anonymization from shared mobile sensor data. In *Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones*, PhoneSense '12, pages 3:1–3:5, New York, NY, USA, 2012. ACM.

[21] Marc Juárez, Sadia Afroz, Gunes Acar, Claudia Díaz, and Rachel Greenstadt. A critical evaluation of website fingerprinting attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 263–274, 2014.

[22] Carmela Troncoso and George Danezis. The bayesian traffic analysis of mix networks. In *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*, pages 369–379, 2009.

[23] Balachander Krishnamurthy and Craig E. Wills. On the leakage of personally identifiable information via online social networks. In *Proceedings of the 2Nd ACM Workshop on Online Social Networks*, WOSN '09, pages 7–12, New York, NY, USA, 2009. ACM.

[24] Arvind Narayanan and Vitaly Shmatikov. Myths and fallacies of "personally identifiable information". *Commun. ACM*, 53(6):24–26, June 2010.

[25] Paul M. Schwartz and Daniel J. Solove. The PII Problem: Privacy and a New Concept of Personally Identifiable Information. *NYUL Rev.*, 86:1814–1894, 2011.

[26] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the 7th International Conference on Database Theory*, ICDT '99, pages 217–235, London, UK, UK, 1999. Springer-Verlag.

[27] Kumar Sharad and George Danezis. De-anonymizing d4d datasets. In *6th Workshop on Hot Topics in Privacy Enhancing Technologies*, HotPETs '13, 2013.

[28] Scott E. Coull, Fabian Monrose, Michael K. Reiter, and Michael Bailey. The challenges of effectively anonymizing network data. In *Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security*, CATCH '09, pages 230–236, Washington, DC, USA, 2009. IEEE Computer Society.

[29] Elena Zheleva and Lise Getoor. To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 531–540, New York, NY, USA, 2009. ACM.

[30] Monica Chew, Dirk Balfanz, and Ben Laurie. (Under)mining privacy in social networks. 2008.

[31] Adrienne Felt and David Evans. Privacy protection for social networking apis. *Workshop on Web 2.0 Security and Privacy (W2SP), Oakland, CA*, 2008.

[32] Matthew M. Lucas and Nikita Borisov. Flybynight: Mitigating the privacy risks of social networking. In *Proceedings of the 7th ACM Workshop on Privacy in the Electronic Society*, WPES '08, pages 1–8, New York, NY, USA, 2008. ACM.

[33] Kapil Singh, Sumeer Bhola, and Wenke Lee. xbook: Redesigning privacy control in social networking platforms. In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM'09, pages 249–266, Berkeley, CA, USA, 2009. USENIX Association.

[34] Florian Kerschbaum and Andreas Schaad. Privacy-preserving social network analysis for criminal investigations. In *Proceedings of the 7th ACM Workshop on Privacy in the Electronic Society*, WPES '08, pages 9–14, New York, NY, USA, 2008. ACM.

[35] Keith B. Frikken and Philippe Golle. Private social network analysis: How to assemble pieces of a graph privately. In *Proceedings of the 5th ACM Workshop on Privacy in Electronic Society*, WPES '06, pages 89–98, New York, NY, USA, 2006. ACM.

[36] Aleksandra Korolova, Rajeev Motwani, Shubha U. Nabar, and Ying Xu. Link privacy in social networks. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, pages 289–298, 2008.

[37] Latanya Sweeney. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, October 2002.

[38] Roberto J. Bayardo Jr. and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, 5-8 April 2005, Tokyo, Japan*, pages 217–228, 2005.

[39] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), March 2007.

[40] Traian Marius Truta and Bindu Vinay. Privacy protection: p-sensitive k-anonymity property. In *Proceedings of the 22nd International Conference on Data Engineering Workshops, ICDE 2006, 3-7 April 2006, Atlanta, GA, USA*, page 94, 2006.

[41] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 106–115, 2007.

[42] Josep Domingo-Ferrer and Vicenç Torra. A critique of k-anonymity and some of its enhancements. In *Proceedings of the 2008 Third International Conference on Availability, Reliability and Security*, ARES '08, pages 990–993, Washington, DC, USA, 2008. IEEE Computer Society.

[43] Xintao Wu, Xiaowei Ying, Kun Liu, and Lei Chen. A survey of privacy-preservation of graphs and social networks. In Charu C. Aggarwal and Haixun Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 421–453. Springer US, 2010.

[44] Elena Zheleva and Lise Getoor. Privacy in social networks: A survey. In Charu C. Aggarwal, editor, *Social Network Data Analytics*, pages 277–306. Springer US, 2011.

[45] Bin Zhou, Jian Pei, and WoShun Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explor. Newsl.*, 10(2):12–22, December 2008.

[46] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pages 1–12, 2006.

[47] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pages 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.

[48] Ashwin Machanavajjhala and Daniel Kifer. Designing statistical privacy for your data. *Commun. ACM*, 58(3):58–67, February 2015.

[49] Miguel E. Andrés, Nicolás Emilio Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: differential privacy for location-based systems. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 901–914, 2013.

[50] Rui Chen, Gergely Acs, and Claude Castelluccia. Differentially private sequential data publication via variable-length n-grams. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 638–649, New York, NY, USA, 2012. ACM.

[51] Cynthia Dwork. Differential privacy: A survey of results. In *Proceedings of the 5th International Conference on Theory and Applications of Models of Computation*, TAMC'08, pages 1–19, Berlin, Heidelberg, 2008. Springer-Verlag.

[52] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 371–380, 2009.

[53] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 715–724, New York, NY, USA, 2010. ACM.

[54] Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 51–60, Washington, DC, USA, 2010. IEEE Computer Society.

[55] Arik Friedman and Assaf Schuster. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 493–502, New York, NY, USA, 2010. ACM.

[56] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. Optimizing linear counting queries under differential privacy. In *Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '10, pages 123–134, New York, NY, USA, 2010. ACM.

[57] Frank McSherry and Ilya Mironov. Differentially private recommender systems: Building privacy into the net. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 627–636, New York, NY, USA, 2009. ACM.

[58] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '07, pages 94–103, Washington, DC, USA, 2007. IEEE Computer Society.

[59] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil P. Vadhan. Computational differential privacy. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 126–142, 2009.

[60] Xiaojian Zhang, Xiaofeng Meng, and Rui Chen. Differentially private set-valued data release against incremental updates. In *Database Systems for Advanced Applications,*

18th International Conference, DASFAA 2013, Wuhan, China, April 22-25, 2013. Proceedings, Part I, pages 392–406, 2013.

[61] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 193–204, New York, NY, USA, 2011. ACM.

[62] Daniel Kifer and Ashwin Machanavajjhala. A rigorous and customizable framework for privacy. In *Proceedings of the 31st Symposium on Principles of Database Systems*, PODS '12, pages 77–88, New York, NY, USA, 2012. ACM.

[63] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 87–96, New York, NY, USA, 2013. ACM.

[64] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 75–84, New York, NY, USA, 2007. ACM.

[65] Vibhor Rastogi, Michael Hay, Gerome Miklau, and Dan Suciu. Relationship privacy: Output perturbation for queries with joins. In *Proceedings of the Twenty-eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '09, pages 107–116, New York, NY, USA, 2009. ACM.

[66] Vishesh Karwa and Aleksandra B. Slavković. Differentially private graphical degree sequences and synthetic graphs. In *Proceedings of the 2012 International Conference on Privacy in Statistical Databases*, PSD'12, pages 273–285, Berlin, Heidelberg, 2012. Springer-Verlag.

[67] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *ACM Trans. Database Syst.*, 39(3):22:1–22:33, October 2014.

[68] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y. Zhao. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 81–98, New York, NY, USA, 2011. ACM.

[69] Yue Wang and Xintao Wu. Preserving differential privacy in degree-correlation based graph generation. *Trans. Data Privacy*, 6(2):127–145, August 2013.

[70] Qian Xiao, Rui Chen, and Kian-Lee Tan. Differentially private network data release via structural inference. In *Proceedings of the 20th ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining, KDD '14, pages 911–920, New York, NY, USA, 2014. ACM.

[71] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, ICDM '09, pages 169–178, Washington, DC, USA, 2009. IEEE Computer Society.

[72] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Proceedings of the 10th Theory of Cryptography Conference on Theory of Cryptography*, TCC'13, pages 457–476, Berlin, Heidelberg, 2013. Springer-Verlag.

[73] Davide Proserpio, Sharon Goldberg, and Frank McSherry. A workflow for differentially-private graph synthesis. In *Proceedings of the 2012 ACM Workshop on Workshop on Online Social Networks*, WOSN '12, pages 13–18, New York, NY, USA, 2012. ACM.

[74] Davide Proserpio, Sharon Goldberg, and Frank McSherry. Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets. *Proc. VLDB Endow.*, 7(8):637–648, April 2014.

[75] Johannes Gehrke, Edward Lui, and Rafael Pass. Towards privacy for social networks: A zero-knowledge based definition of privacy. In *Proceedings of the 8th Conference on Theory of Cryptography*, TCC'11, pages 432–449, Berlin, Heidelberg, 2011. Springer-Verlag.

[76] Yves-Alexandre de Montjoye, Erez Shmueli, Samuel S. Wang, and Alex Sandy Pentland. openpds: Protecting the privacy of metadata through safeanswers. *PLoS ONE*, 9(7):1–9, 07 2014.

[77] Elena Zheleva and Lise Getoor. Preserving the privacy of sensitive relationships in graph data. In *Proceedings of the 1st ACM SIGKDD International Conference on Privacy, Security, and Trust in KDD*, PinKDD'07, pages 153–171, Berlin, Heidelberg, 2008. Springer-Verlag.

[78] Graham Cormode, Divesh Srivastava, Ting Yu, and Qing Zhang. Anonymizing bipartite graph data using safe groupings. *Proc. VLDB Endow.*, 1(1):833–844, August 2008.

[79] Michael Hay, Gerome Miklau, David Jensen, Don Towsley, and Philipp Weis. Resisting structural re-identification in anonymized social networks. *Proc. VLDB Endow.*, 1(1):102–114, August 2008.

[80] Alina Campan and Traian Marius Truta. Data and structural k-anonymity in social networks. In *Privacy, Security, and Trust in KDD, Second ACM SIGKDD International Workshop, PinKDD 2008, Las Vegas, NV, USA, August 24, 2008, Revised Selected Papers*, pages 33–54, 2008.

[81] Smriti Bhagat, Graham Cormode, Balachander Krishnamurthy, and Divesh Srivastava. Class-based graph anonymization for social network data. *Proc. VLDB Endow.*, 2(1):766–777, August 2009.

[82] Francesco Bonchi, Aristides Gionis, and Tamir Tassa. Identity obfuscation in graphs through the information theoretic lens. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ICDE '11, pages 924–935, Washington, DC, USA, 2011. IEEE Computer Society.

[83] Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. Anonymizing social networks. *Computer Science Department Faculty Publication Series*, page 180, 2007.

[84] Xiaowei Ying and Xintao Wu. Randomizing social networks: a spectrum preserving approach. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, USA*, pages 739–750, 2008.

[85] Xiaowei Ying and Xintao Wu. Graph generation with prescribed feature constraints. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*, pages 966–977, 2009.

[86] Xiaowei Ying and Xintao Wu. On link privacy in randomizing social networks. In *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '09, pages 28–39, Berlin, Heidelberg, 2009. Springer-Verlag.

[87] Lars Backstrom, Cynthia Dwork, and Jon M. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 181–190, 2007.

[88] Lian Liu, Jie Wang, Jinze Liu, and Jun Zhang. Privacy preserving in social networks against sensitive edge disclosure. Technical report, Technical Report Technical Report CMIDA-HiPSCCS 006-08, Department of Computer Science, University of Kentucky, KY, 2008.

[89] Mingqiang Xue, Panagiotis Karras, Raissi Chedy, Panos Kalnis, and Hung Keng Pung. Delineating social network data anonymization via random edge perturbation. In *Proceedings of the 21st ACM International Conference on Information*

and Knowledge Management, CIKM '12, pages 475–484, New York, NY, USA, 2012. ACM.

[90] Prateek Mittal, Charalampos Papamanthou, and Dawn Xiaodong Song. Preserving link privacy in social network based systems. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*, 2013.

[91] Changchang Liu and Prateek Mittal. Linkmirage: Enabling privacy-preserving analytics on social relationships. In *23nd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*, 2016.

[92] Ninghui Li, Wahbeh H. Qardaji, and Dong Su. Provably private data anonymization: Or, k-anonymity meets differential privacy. *CoRR*, abs/1101.2604, 2011.

[93] James Cheng, Ada Wai-chee Fu, and Jia Liu. K-isomorphism: Privacy preserving network publication against structural attacks. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 459–470, New York, NY, USA, 2010. ACM.

[94] Lei Zou, Lei Chen, and M. Tamer Özsu. K-automorphism: A general framework for privacy preserving network publication. *Proc. VLDB Endow.*, 2(1):946–957, August 2009.

[95] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 93–106, New York, NY, USA, 2008. ACM.

[96] Bin Zhou and Jian Pei. The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowledge and Information Systems*, 28(1):47–77, 2011.

[97] Brian Thompson and Danfeng Yao. The union-split algorithm and cluster-based anonymization of social networks. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS '09, pages 218–227, New York, NY, USA, 2009. ACM.

[98] Wentao Wu, Yanghua Xiao, Wei Wang, Zhenying He, and Zhihui Wang. K-symmetry model for identity anonymization in social networks. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 111–122, New York, NY, USA, 2010. ACM.

[99] Rudolf Mathon. A note on the graph isomorphism counting problem. *Inf. Process. Lett.*, 8(3):131–132, 1979.

[100] Alket Cecaj, Marco Mamei, and Nicola Bicocchi. Re-identification of anonymized CDR datasets using social network data. In *2014 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom 2014 Workshops, Budapest, Hungary, March 24-28, 2014*, pages 237–242, 2014.

[101] Aston Zhang, Xing Xie, Kevin Chen-Chuan Chang, Carl A. Gunter, Jiawei Han, and XiaoFeng Wang. Privacy risk in anonymized heterogeneous information networks. In *Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014.*, pages 595–606, 2014.

[102] Shouling Ji, Weiqing Li, Prateek Mittal, Xin Hu, and Raheem Beyah. Secgraph: A uniform and open-source evaluation system for graph data anonymization and de-anonymization. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 303–318, Washington, D.C., August 2015. USENIX Association.

[103] Shouling Ji, Weiqing Li, Mudhakar Srivatsa, Jing (Selena) He, and Raheem A. Beyah. Structure based data de-anonymization of social networks and mobility traces. In *Information Security - 17th International Conference, ISC 2014, Hong Kong, China, October 12-14, 2014. Proceedings*, pages 237–254, 2014.

[104] Shouling Ji, Weiqing Li, Mudhakar Srivatsa, and Raheem Beyah. Structural data de-anonymization: Quantification, practice, and implications. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1040–1053, New York, NY, USA, 2014. ACM.

[105] Vicenç Torra and Klara Stokes. A formalization of re-identification in terms of compatible probabilities. *CoRR*, abs/1301.5022, 2013.

[106] Shirin Nilizadeh, Apu Kapadia, and Yong-Yeol Ahn. Community-enhanced de-anonymization of online social networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 537–548, New York, NY, USA, 2014. ACM.

[107] Mudhakar Srivatsa and Mike Hicks. Deanonymizing mobility traces: Using social network as a side-channel. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 628–637, New York, NY, USA, 2012. ACM.

[108] Lorenzo Alvisi, Allen Clement, Alessandro Epasto, Silvio Lattanzi, and Alessandro Panconesi. Sok: The evolution of sybil defense via social networks. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 382–396, 2013.

[109] George Danezis and Prateek Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2009, San Diego, California, USA, 8th February - 11th February 2009*, 2009.

[110] Manuel Egele, Gianluca Stringhini, Christopher Krügel, and Giovanni Vigna. COMPA: detecting compromised accounts on social networks. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*, 2013.

[111] Bimal Viswanath, Ansley Post, Krishna P. Gummadi, and Alan Mislove. An analysis of social network-based sybil defenses. *SIGCOMM Comput. Commun. Rev.*, 40(4):363–374, August 2010.

[112] Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. *IEEE/ACM Trans. Netw.*, 18(3):885–898, June 2010.

[113] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. Sybilguard: Defending against sybil attacks via social networks. *SIGCOMM Comput. Commun. Rev.*, 36(4):267–278, August 2006.

[114] Nitish Korula and Silvio Lattanzi. An efficient reconciliation algorithm for social networks. *Proc. VLDB Endow.*, 7(5):377–388, January 2014.

[115] Lyudmila Yartseva and Matthias Grossglauser. On the performance of percolation graph matching. In *Proceedings of the First ACM Conference on Online Social Networks*, COSN '13, pages 119–130, New York, NY, USA, 2013. ACM.

[116] Shouling Ji, Weiqing Li, Neil Zhenqiang Gong, Prateek Mittal, and Raheem A. Beyah. On your social network de-anonymizablity: Quantification and large scale evaluation with seed knowledge. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2014*, 2015.

[117] Gilbert Wondracek, Thorsten Holz, Engin Kirda, and Christopher Kruegel. A practical attack to de-anonymize social network users. In *31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berleley/Oakland, California, USA*, pages 223–238, 2010.

[118] Pedram Pedarsani, Daniel R. Figueiredo, and Matthias Grossglauser. A bayesian method for matching two similar graphs without seeds. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing, Allerton Park & Retreat Center, Monticello, IL, USA, October 2-4, 2013*, pages 1598–1607, 2013.

[119] Pedram Pedarsani and Matthias Grossglauser. On the privacy of anonymized networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 1235–1243, New York, NY, USA, 2011. ACM.

[120] Yoni De Mulder, George Danezis, Lejla Batina, and Bart Preneel. Identification via location-profiling in GSM networks. In Vijay Atluri and Marianne Winslett, editors, *WPES*, pages 23–32. ACM, 2008.

[121] C. Dwork and M. Naor. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality*, 2(1):8, 2008.

[122] Kumar Sharad and George Danezis. An automated social graph de-anonymization technique. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, pages 47–58, New York, NY, USA, 2014. ACM.

[123] Central Intelligence Agency. CIA - The World Factbook. `https://www.cia.gov/library/publications/the-world-factbook/geos/iv.html`, 2012.

[124] Vincent D. Blondel, Markus Esch, Connie Chan, Fabrice Clérot, Pierre Deville, Etienne Huens, Frédéric Morlot, Zbigniew Smoreda, and Cezary Ziemlicki. Data for development: the D4D challenge on mobile phone data. *CoRR*, abs/1210.0137, 2012.

[125] Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In *Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '04, pages 223–228, New York, NY, USA, 2004. ACM.

[126] D. G. Corneil and C. C. Gotlieb. An efficient algorithm for graph isomorphism. *J. ACM*, 17(1):51–64, January 1970.

[127] László Babai and Eugene M. Luks. Canonical labeling of graphs. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, pages 171–183, New York, NY, USA, 1983. ACM.

[128] László Babai. Graph isomorphism in quasipolynomial time. *arXiv preprint arXiv:1512.03547*, 2015.

[129] Tomek Czajka and Gopal Pandurangan. Improved random graph isomorphism. *J. of Discrete Algorithms*, 6(1):85–92, March 2008.

[130] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, January 1976.

[131] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(10):1367–1372, October 2004.

[132] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recogn. Lett.*, 18(9):689–694, August 1997.

[133] Horst Bunke and Kim Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recogn. Lett.*, 19(3-4):255–259, March 1998.

[134] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42, New York, NY, USA, 2007. ACM.

[135] Ross B. Girshick, Jamie Shotton, Pushmeet Kohli, Antonio Criminisi, and Andrew W. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 415–422, 2011.

[136] Antonio Criminisi, Jamie Shotton, and Stefano Bucciarelli. Decision forests with long-range spatial context for organ localization in ct volumes. In *MICCAI workshop on Probabilistic Models for Medical Image Analysis (MICCAI-PMMIA)*, 2009.

[137] Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(9):1465–1479, 2006.

[138] Grégory Rogez, Jonathan Rihan, Srikumar Ramalingam, Carlos Orrite, and Philip H. S. Torr. Randomized trees for human pose detection. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*, 2008.

[139] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew W. Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Commun. ACM*, 56(1):116–124, 2013.

[140] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[141] Tin Kam Ho. Random decision forests. In *Third International Conference on Document Analysis and Recognition, ICDAR 1995, August 14 - 15, 1995, Montreal, Canada. Volume I*, pages 278–282, 1995.

[142] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[143] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[144] Alexander Zien and Cheng Soon Ong. Multiclass multiple kernel learning. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 1191–1198, 2007.

[145] Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[146] Zhuowen Tu. Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. In *10th IEEE International Conference on Computer Vision (ICCV 2005), 17-20 October 2005, Beijing, China*, pages 1589–1596, 2005.

[147] Pei Yin, Antonio Criminisi, John M. Winn, and Irfan A. Essa. Tree-based classifiers for bilayer video segmentation. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*, 2007.

[148] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2-3):81–227, 2012.

[149] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8):832–844, 1998.

[150] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 556–559, New York, NY, USA, 2003. ACM.

[151] Tsuyoshi Murata and Sakiko Moriyasu. Link prediction based on structural properties of online social networks. *New Generation Comput.*, 26(3):245–257, 2008.

[152] Dan Corlette and Frank M. Shipman, III. Link prediction applied to an open large-scale online social network. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*, HT '10, pages 135–140, New York, NY, USA, 2010. ACM.

[153] Linyuan Lu and Tao Zhou. Link prediction in complex networks: A survey. *CoRR*, abs/1010.0725, 2010.

[154] Benjamin Taskar, Ming Fai Wong, Pieter Abbeel, and Daphne Koller. Link prediction in relational data. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, pages 659–666, 2003.

[155] Matthew Richardson and Pedro M. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[156] Hisashi Kashima and Naoki Abe. A parameterized probabilistic model of network evolution for supervised link prediction. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 340–349, Washington, DC, USA, 2006. IEEE Computer Society.

[157] Mustafa Bilgic, Galileo Namata, and Lise Getoor. Combining collective classification and link prediction. In *Workshops Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*, pages 381–386, 2007.

[158] W. Cukierski, B. Hamner, and Bo Yang. Graph-based features for supervised link prediction. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1237–1244, July 2011.

[159] Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. It's who you know: Graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 663–671, New York, NY, USA, 2011. ACM.

[160] André Nunes, Pável Calado, and Bruno Martins. Resolving user identities over social networks through supervised learning and rich similarity features. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 728–729, New York, NY, USA, 2012. ACM.

[161] R. Zafarani and H. Liu. Social computing data repository at ASU, 2009.

[162] Antonio Criminisi, Jamie Shotton, Duncan P. Robertson, and Ender Konukoglu. Regression forests for efficient anatomy detection and localization in CT studies. In *Medical Computer Vision. Recognition Techniques and Applications in Medical Imaging - International MICCAI Workshop, MCV 2010, Beijing, China, September 20, 2010, Revised Selected Papers*, pages 106–117, 2010.

[163] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*, 2008.

[164] Kumar Sharad. True friends let you down: Benchmarking social graph anonymization schemes. In *Proceedings of the 9th ACM Workshop on Artificial Intelligence and Security*, AISec '16, pages 93–104, New York, NY, USA, 2016. ACM.

[165] Xiaowei Ying, Kai Pan, Xintao Wu, and Ling Guo. Comparisons of randomization and k-degree anonymization schemes for privacy preserving social network publishing. In *Proceedings of the 3rd Workshop on Social Network Mining and Analysis*, SNA-KDD '09, pages 10:1–10:10, New York, NY, USA, 2009. ACM.

[166] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and P. Krishna Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM Workshop on Online Social Networks, WOSN 2009, Barcelona, Spain, August 17, 2009*, pages 37–42, 2009.

[167] L. Singh and J. Zhan. Measuring topological anonymity in social networks. In *Granular Computing, 2007. GRC 2007. IEEE International Conference on*, pages 770–770, Nov 2007.

[168] Gábor György Gulyás and Sándor Imre. Using identity separation against de-anonymization of social networks. *Transactions on Data Privacy*, 8(2):113–140, 2015.

[169] Gábor György Gulyás and Sándor Imre. Analysis of identity separation against a passive clique-based de-anonymization attack. *Infocommunications Journal*, 4(3):11–20, 2011.

[170] Benedek Simon, Gábor György Gulyás, and Sándor Imre. Analysis of grasshopper, a novel social network de-anonymization algorithm. *Periodica Polytechnica. Electrical Engineering and Computer Science*, 58(4):161, 2014.

[171] Gábor György Gulyás and Sándor Imre. Measuring local topological anonymity in social networks. In *12th IEEE International Conference on Data Mining Workshops, ICDM Workshops, Brussels, Belgium, December 10, 2012*, pages 563–570, 2012.

[172] Kumar Sharad. Change of guard: The next generation of social graph de-anonymization attacks. In *Proceedings of the 9th ACM Workshop on Artificial Intelligence and Security*, AISec '16, pages 105–116, New York, NY, USA, 2016. ACM.

[173] Gábor György Gulyás and Sándor Imre. Measuring importance of seeding for structural de-anonymization attacks in social networks. In *2014 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom 2014 Workshops, Budapest, Hungary, March 24-28, 2014*, pages 610–615, 2014.

[174] Karl Bringmann, Tobias Friedrich, and Anton Krohmer. De-anonymization of heterogeneous random graphs in quasilinear time. In *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 197–208, 2014.

[175] Ehsan Kazemi, Seyed Hamed Hassani, and Matthias Grossglauser. Growing a graph matching from a handful of seeds. *PVLDB*, 8(10):1010–1021, 2015.

List of figures

2.1 Random Switch 25

2.2 The 2-hop egonet of a node 34

3.1 Scheme 1 preserves the complete 2-hop network while Scheme 2 removes edges between 2-hop nodes 41

3.2 The model for D4D sub-net linking learning task 45

3.3 Example node feature vector 49

3.4 A randomly trained decision tree. The split nodes store weak learners $\delta(v_p[i], v_q[j]) \leq \tau$, the highlighted leaf node has posterior of $(0.24, 0.76)$ 51

3.5 Epinions (self-validation): ROC curves for both schemes 54

3.6 Pokec (self-validation): ROC curves for both schemes 55

3.7 Pokec (x-validation): ROC curves for both schemes 57

3.8 Epinions (x-validation): ROC curves for both schemes 58

3.9 Flickr: ROC curves for edge perturbation 60

3.10 True Positive and False Positive rates vary widely depending on the social neighborhood overlap 61

3.11 Epinions (Scheme 2, complete): ROC curves for effect of vector length 67

4.1 Example node feature vector 75

4.2 Original joint degree distribution 78

4.3 RSP: ROC curves 79

4.4 RSP: Degree Distribution 80

4.5 RSP: Joint Degree Distribution 80

4.6 RSP: Degree Connectivity vs. Node Degree 81

4.7 RSP: Eigenvector Centrality vs. Degree Centrality 81

4.8 RAD: ROC curves 82
4.9 RAD: Degree Distribution 83
4.10 RAD: Joint Degree Distribution 83
4.11 RAD: Degree Connectivity vs. Node Degree 84
4.12 RAD: Eigenvector Centrality vs. Degree Centrality 84
4.13 RSW: ROC curves 85
4.14 RSW: Joint Degree Distribution 85
4.15 RSW: Degree Connectivity vs. Node Degree 86
4.16 RSW: Eigenvector Centrality vs. Degree Centrality 86
4.17 REP: ROC curves 87
4.18 REP: Degree Distribution 88
4.19 REP: Joint Degree Distribution 88
4.20 REP: Degree Connectivity vs. Node Degree 89
4.21 REP: Eigenvector Centrality vs. Degree Centrality 89
4.22 KDA: ROC curves 90
4.23 KDA: Degree Distribution 90
4.24 KDA: Joint Degree Distribution 91
4.25 KDA: Degree Connectivity vs. Node Degree 91
4.26 KDA: Eigenvector Centrality vs. Degree Centrality 92
4.27 1HKA: ROC curves 92
4.28 1HKA: Degree Distribution 93
4.29 1HKA: Joint Degree Distribution 93
4.30 1HKA: Degree Connectivity vs. Node Degree 94
4.31 1HKA: Eigenvector Centrality vs. Degree Centrality 94
4.32 Scheme Comparison: ROC curves 101

5.1 Feature vector of quantized node neighborhood 109
5.2 The ROC curve (AUC in legend) 114
5.3 Success of adversary 116
5.4 Variance of accuracy and coverage with degree 117
5.5 Variance of accuracy and coverage with classification threshold 118

5.6 Variance of accuracy and coverage on iterating 119

5.7 Variance of mapping count on iterating 120

5.8 Joint Degree Distribution: True mappings 121

5.9 Joint Degree Distribution: False mappings 122

5.10 Joint Degree Distribution: Unidentified mappings 123

5.11 Flickr ($\alpha_E = \alpha_V = 0.33$) 123

5.12 Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$) 124

5.13 Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$) 124

5.14 Enron ($\alpha_E = 0.43, \alpha_V = 1$) 125

5.15 Cumulative percentage vs Jaccard Coefficient 126

List of tables

3.1 Success percentage for ad-hoc de-anonymization of Scheme 1 for Case 1 node pairs 53

3.2 Epinions (self-validation): *False Positive* vs. *True Positive* for both schemes 55

3.3 Pokec (self-validation): *False Positive* vs. *True Positive* for both schemes 56

3.4 Pokec (x-validation): *False Positive* vs. *True Positive* for both schemes 57

3.5 Epinions (x-validation): *False Positive* vs. *True Positive* for both schemes 58

3.6 Flickr (edge perturbation): *False Positive* vs. *True Positive* 60

3.7 Scheme 1: Sample sizes for ad-hoc de-anonymization of Case 1 node pairs 62

3.8 Training and testing number of pairs for Epinions and Pokec 62

3.9 Training and testing number of pairs for Flickr 62

3.10 Training and testing times for Epinions and Pokec 63

3.11 Training and testing times for Flickr 63

3.12 Epinions: Varying vector length (Scheme 2 - Complete): *False Positive* vs. *True Positive* 68

4.1 Flickr: *False Positive* vs. *True Positive* 99

4.2 Facebook: *False Positive* vs. *True Positive* 100

4.3 Hellinger Distance between degree distribution (DD) and joint degree distribution (JDD) for perturbed and unperturbed graphs and its effect on ROC curve's Area Under the Curve (AUC) 102

5.1 Graph details 112

5.2 Common nodes 112

5.3 Degree thresholds for all phases 112

5.4 Number of identical (**I**) and non-identical (**NI**) samples used for training 113

5.5 Phase 1: *False Positive* vs. *True Positive* 114

5.6 Number of mappings produced 114

5.7 Mapping accuracy (%) 115

5.8 Mapping coverage for node degree above t_3 (%) 115

5.9 Comparison of coverage and accuracy percentage with other attacks (accuracy = coverage for all attacks except `3PSL`); $^{\varnothing}$ denotes a seedless attack. A higher percentage is better. 127